

**数据库系统概论实验报告**

|  |  |
| --- | --- |
|  |  |
| 姓 名： | 万祖桓 |
| 学 院： | 网络空间安全学院 |
| 专 业： | 网络空间安全 |
| 班 级： | 网安2203班 |
| 学 号： | U202212004 |
| 指导教师： | 路松峰 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2023 年 12 月 29 日

目录

[1 课程任务概述 4](#_Toc154846902)

[2数据库定义与基本操作 5](#_Toc154846903)

[2.1 任务要求 5](#_Toc154846904)

[2.2 完成过程 5](#_Toc154846905)

[2.3任务总结 9](#_Toc154846915)

[3 SQL 的复杂操作 10](#_Toc154846916)

[3.1 任务要求 10](#_Toc154846917)

[3.2 完成过程 10](#_Toc154846918)

[3.3任务总结 15](#_Toc154846929)

[4 实验任务三（标题改为具体任务名称） 16](#_Toc154846930)

[4.1 任务要求 16](#_Toc154846931)

[4.2 完成过程 17](#_Toc154846932)

[4.3任务总结 24](#_Toc154846946)

[5 实验任务四（标题改为具体任务名称） 26](#_Toc154846947)

[5.1 任务要求 26](#_Toc154846948)

[5.2 完成过程 26](#_Toc154846949)

[（1）实现步骤： 27](#_Toc154846950)

[5.3任务总结 50](#_Toc154846951)

[6 课程总结 51](#_Toc154846952)

[附录 52](#_Toc154846953)

# 1 课程任务概述

实验一：数据库定义与基本操作

1. 掌握 DBMS 的数据定义功能
2. 掌握 SQL 语言的数据定义语句
3. 掌握 DBMS 的数据单表查询功能
4. 掌握 SQL 语言的数据单表查询语句

实验二：SQL 的复杂操作

掌握 SQL 语言的数据多表查询语句和更新操作

实验三：SQL 的高级实验

掌握 SQL 语言的视图、触发器、存储过程、安全等功能

实验四：数据库设计

掌握数据库设计和开发技巧

# 2数据库定义与基本操作

## 2.1 任务要求

（1）查询全体学生的学号、姓名和年龄；

（2）查询所有计算机系学生的详细记录；

（3）找出考试成绩为优秀（90 分及以上）或不及格的学生的学号、课程号及成绩；

（4）查询年龄不在 19~20 岁之间的学生姓名、性别和年龄；

（5）查询数学系（MA）、信息系（IS)的学生的姓名和所在系；

（6）查询名称中包含“数据”的所有课程的课程号、课程名及其学分；

（7) 找出所有没有选修课成绩的学生学号和课程号；

（8）查询学生 200215121 选修课的最高分、最低分以及平均成绩；

（9）查询选修了 2 号课程的学生的学号及其成绩，查询结果按成绩升序排列；

（10）查询每个系名及其学生的平均年龄。

（思考：如何查询学生平均年龄在 19 岁以下（含 19 岁）的系别及其学生的平均年龄？）

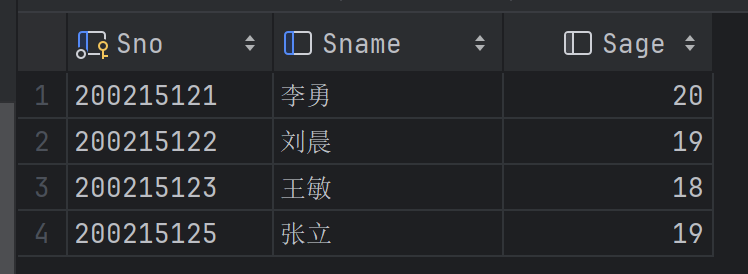
## 2.2 完成过程

### 2.2.1 （1）查询全体学生的学号、姓名和年龄；

代码：

select Sno,Sname,Sage from student;

运行结果截图（使用datagrip）：

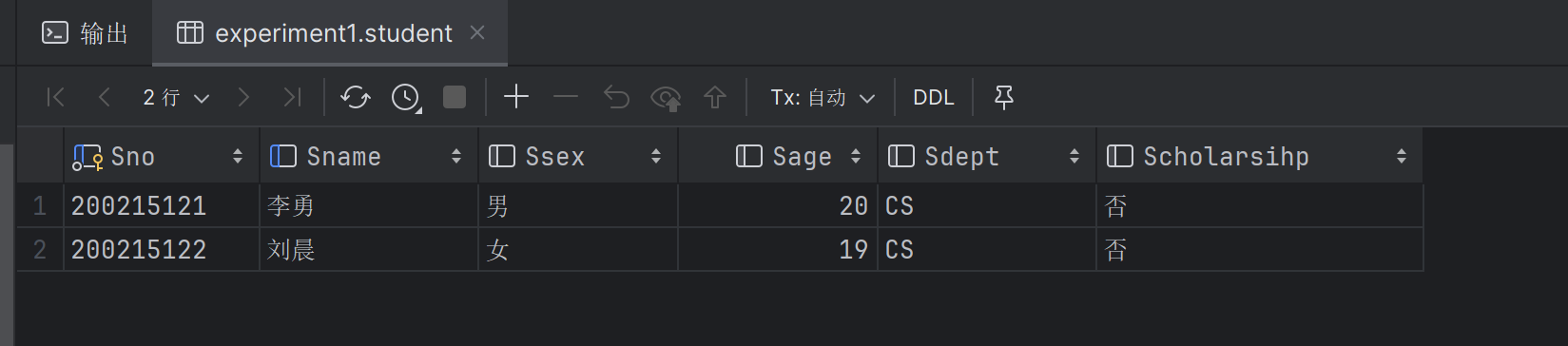


### 2.2.2（2）查询所有计算机系学生的详细记录；

代码：

select \* from student where Sdept='CS';

运行结果截图：

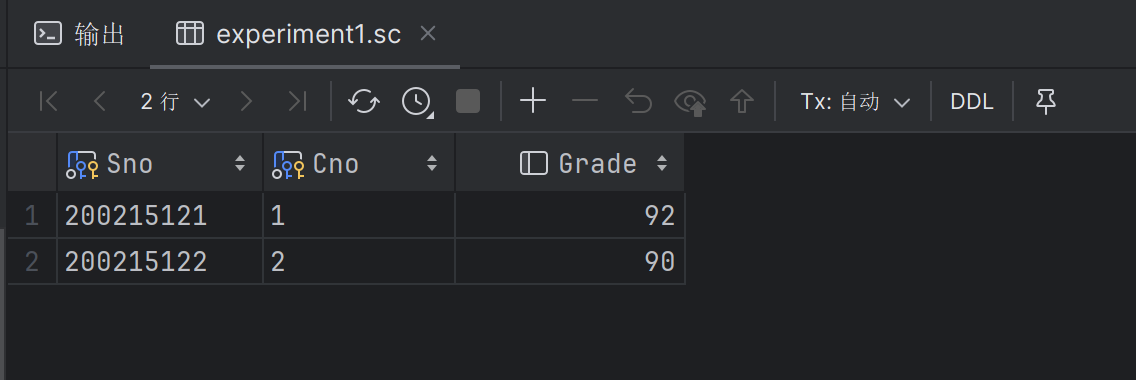


### 2.2.3（3）找出考试成绩为优秀（90 分及以上）或不及格的学生的学号、课程号及成绩；

代码：

select Sno,Cno,Grade from SC where Grade>=90||Grade<60;

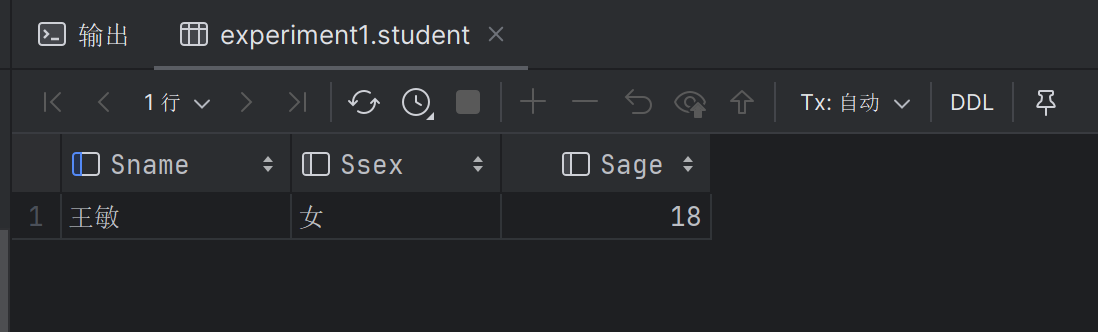
运行结果截图

**2.2.4（4）查询年龄不在 19~20 岁之间的学生姓名、性别和年龄；**

代码：

select Sname,Ssex,Sage from student where Sage<19||Sage>20;

运行结果截图：

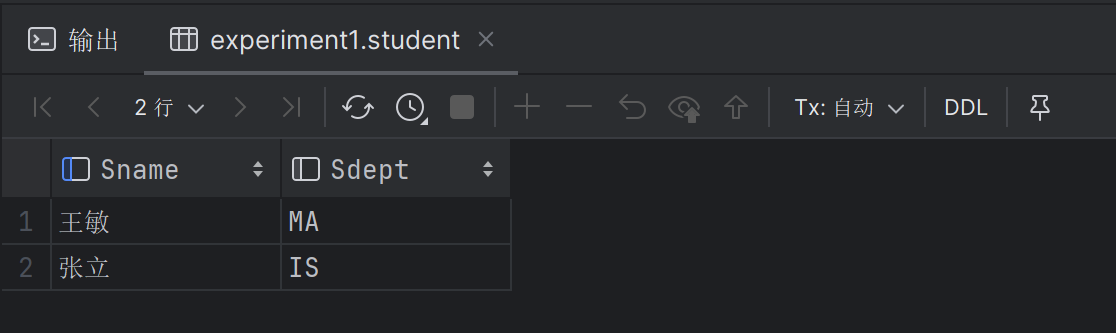


### 2.2.5（5）查询数学系（MA）、信息系（IS)的学生的姓名和所在系；

代码：

select Sname,Sdept from student where Sdept='MA'||Sdept='IS';

运行结果截图：

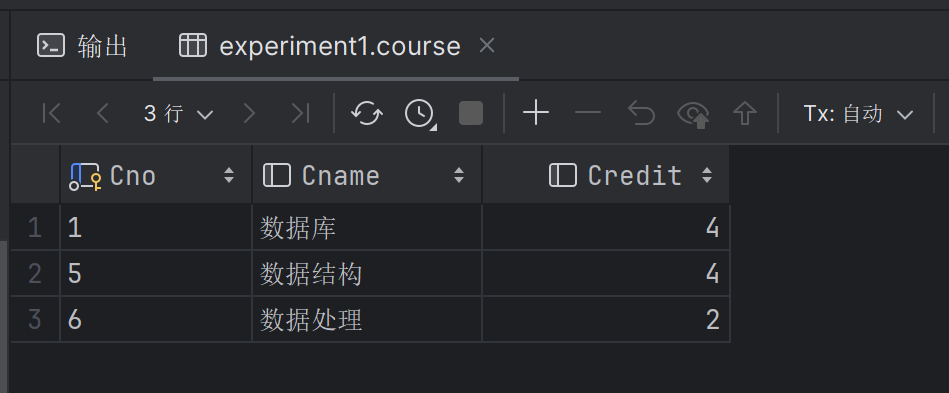


### 2.2.6（6）查询名称中包含“数据”的所有课程的课程号、课程名及其学分；

代码：

select Cno,Cname,Credit from course where Cname like '%数据%';

运行结果截图：

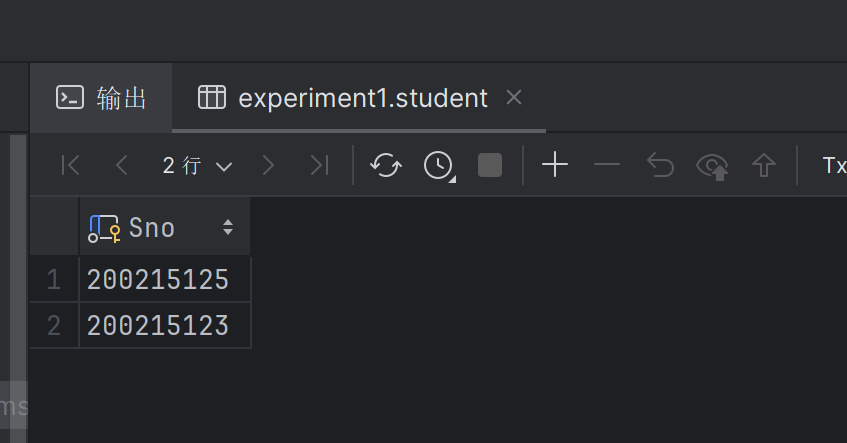


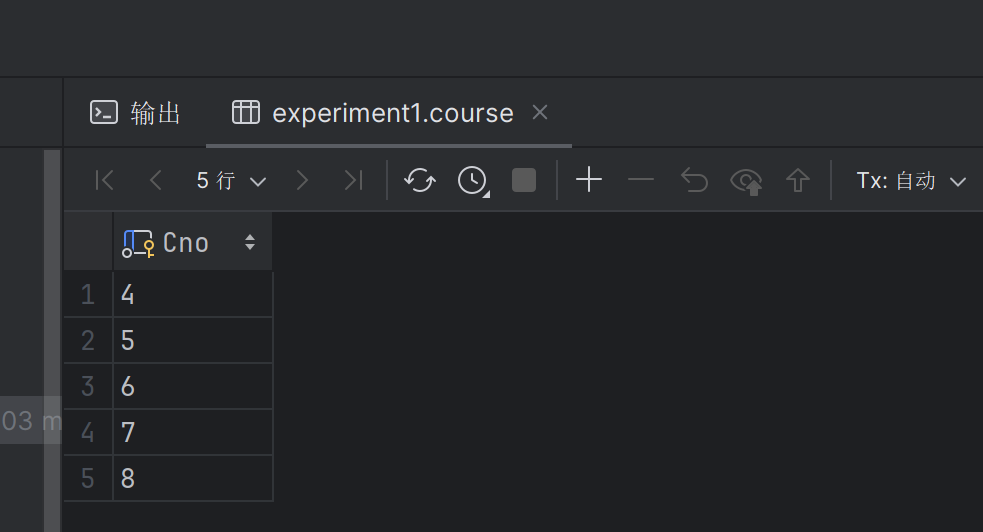
### 2.2.7（7） 找出所有没有选修课成绩的学生学号和课程号；

代码：

select Sno from student where Sno not in (select Sno from sc);  
select Cno from course where Cno not in (select Cno from sc);

运行结果截图：



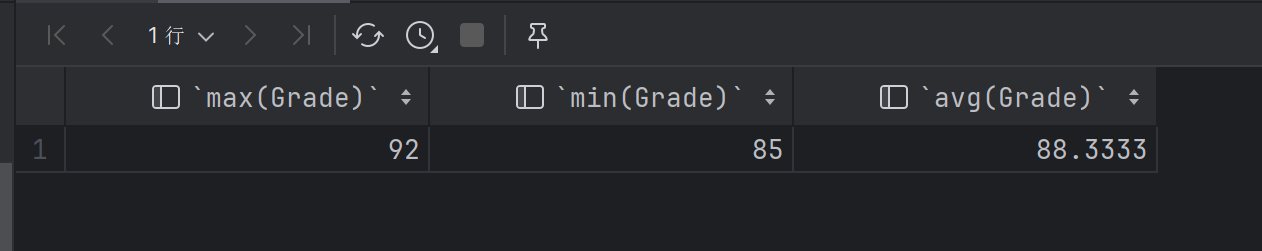


### 2.2.8（8）查询学生 200215121 选修课的最高分、最低分以及平均成绩；

代码：

select *max*(Grade),*min*(Grade),*avg*(Grade) from SC where Sno='200215121';

运行结果截图：

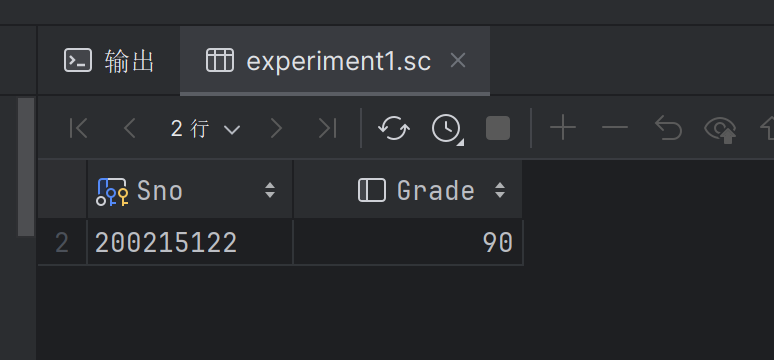
****

### 2.2.9（9）查询选修了 2 号课程的学生的学号及其成绩，查询结果按成绩升序排列；

代码：

select Sno,Grade from sc where Cno='2' order by Grade asc ;

运行结果截图：

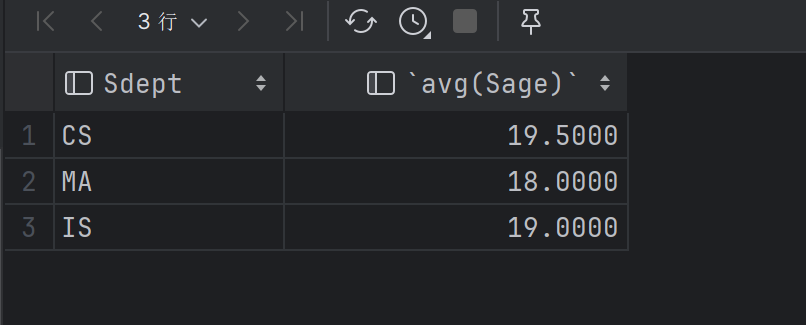


### 2.2.10（10）查询每个系名及其学生的平均年龄。

代码：

select Sdept,*avg*(Sage) from student group by Sdept;

运行结果截图：



## 2.3任务总结

拓展任务难度不大，在做（7）时一开始没有思路，后借助搜索引擎，复习了一下not in子句，顺利完成。这次实验让我对SQL语言有了基本的认识，了解了DBMS的概念，掌握了简单的SQL数据表单查询语句。

# 3 SQL 的复杂操作

## 3.1 任务要求

（1）查询每门课程及其被选情况（输出所有课程中每门课的课程号、课程名称、选修该课程的学生

学号及成绩--如果没有学生选择该课，则相应的学生学号及成绩为空值）。

（2）查询与“张立”同岁的学生的学号、姓名和年龄。（要求使用至少 3 种方法求解）

（3）查询选修了 3 号课程而且成绩为良好（80~89 分）的所有学生的学号和姓名。

（4）查询学生 200215122 选修的课程号、课程名

（思考：如何查询学生 200215122 选修的课程号、课程名及成绩？）

（5）找出每个学生低于他所选修课程平均成绩 5 分以上的课程号。（输出学号和课程号）

（6）查询比所有男生年龄都小的女生的学号、姓名和年龄。

（7）查询所有选修了 2 号课程的学生姓名及所在系。

（8）使用 update 语句把成绩为良的学生的年龄增加 2 岁，并查询出来。

（9）使用 insert 语句增加两门课程：C 语言和人工智能，并查询出来

（10）使用 delete 语句把人工智能课程删除，并查询出来。

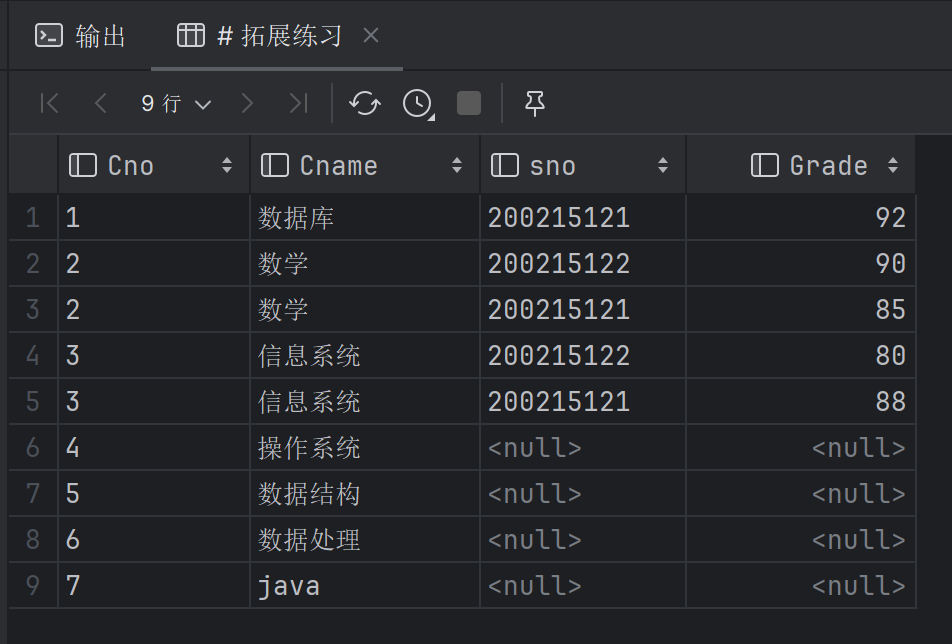
## 3.2 完成过程

### 3.2.1（1）查询每门课程及其被选情况（输出所有课程中每门课的课程号、课程名称、选修该课程的学生

代码：

select course.Cno,Cname,sc.sno,Grade from course left join sc on course.Cno=sc.Cno;

运行结果截图：

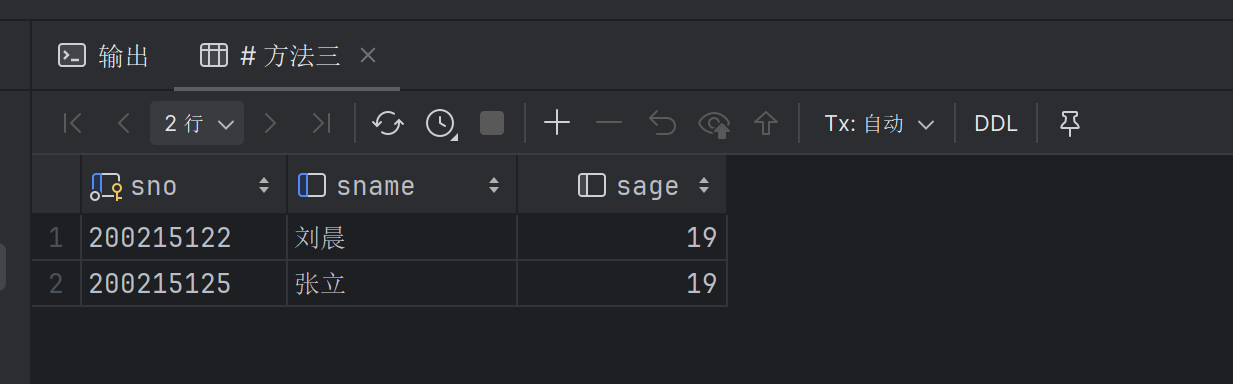


### 3.2.2（2）查询与“张立”同岁的学生的学号、姓名和年龄。（要求使用至少 3 种方法求解）

代码：

# 方法一  
select student.sno,Sname,Sage from student where Sage=(  
 select Sage from student where Sname='张立'  
 );  
# 方法二  
select student.sno,Sname,Sage from student where Sage in (  
 select Sage from student where Sname='张立'  
 );  
# 方法三  
select s1.sno,s1.sname,s1.sage from student s1,student s2 where s1.Sage=s2.Sage and s2.Sname='张立';

运行结果截图：

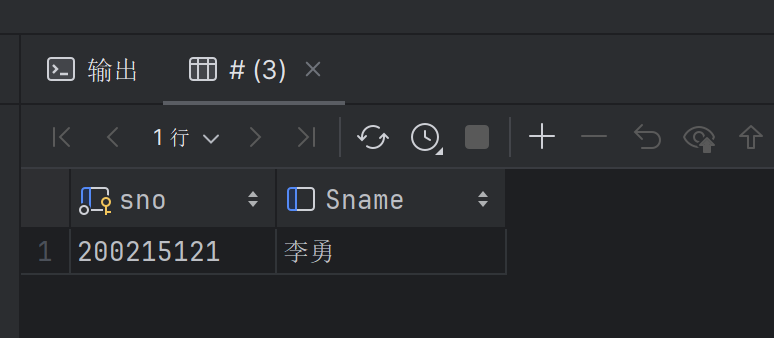


### 3.2.3（3）查询选修了 3 号课程而且成绩为良好（80~89 分）的所有学生的学号和姓名。

代码：

select student.sno,Sname from student,sc where student.sno=sc.sno and sc.Cno='2' and sc.Grade between 80 and 89;

运行结果截图：

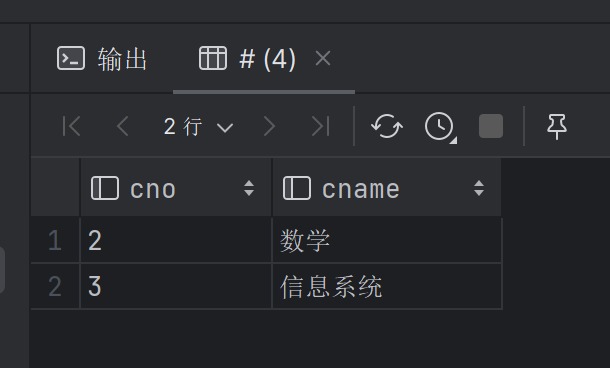


### 3.2.4（4）查询学生 200215122 选修的课程号、课程名

代码：

select sc.cno,course.cname from sc,course where sc.Sno='200215122' and sc.Cno=course.Cno;

运行结果截图：

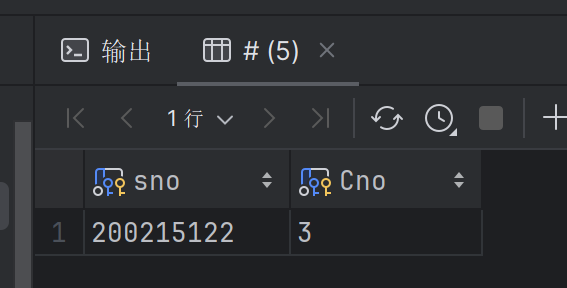


### 3.2.5（5）找出每个学生低于他所选修课程平均成绩 5 分以上的课程号。（输出学号和课程号）

代码:

select sc.sno,Cno from sc where (select *avg*(Grade) from sc)-Grade>=5;

运行结果截图：

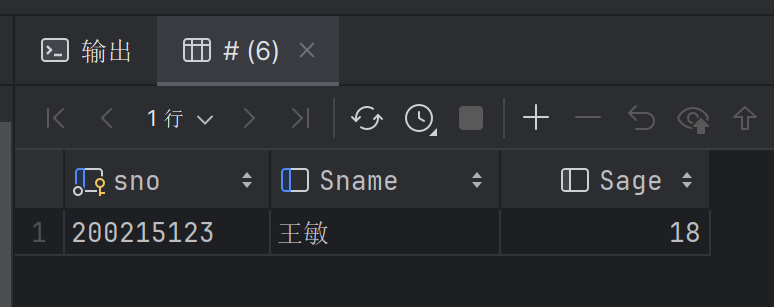


### 3.2.6（6）查询比所有男生年龄都小的女生的学号、姓名和年龄。

代码：

select student.sno,Sname,Sage from student where Ssex='女' and Sage<(select *min*(Sage) from student where Ssex='男');

运行结果截图：

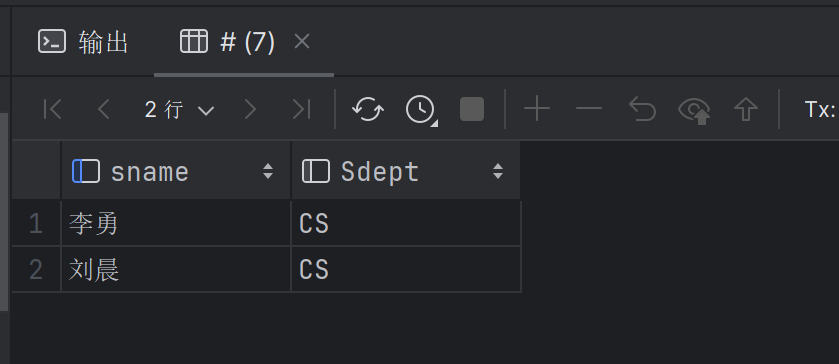


### 3.2.7（7）查询所有选修了 2 号课程的学生姓名及所在系。

代码：

select student.sname,Sdept from student,sc where student.sno=sc.Sno and sc.Cno='2';

实验运行结果：

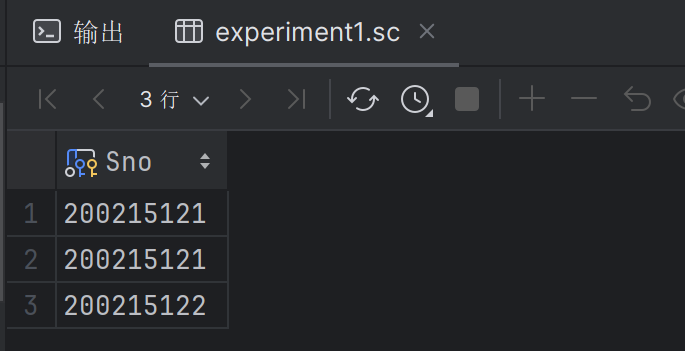


### 3.2.8（8）使用 update 语句把成绩为良的学生的年龄增加 2 岁，并查询出来。

代码：

update student set Sage=Sage+2 where Sno in(select sc.Sno from sc where sc.Grade between 80 and 89);  
select sc.Sno from sc where Grade between 80 and 89;

运行结果截图：

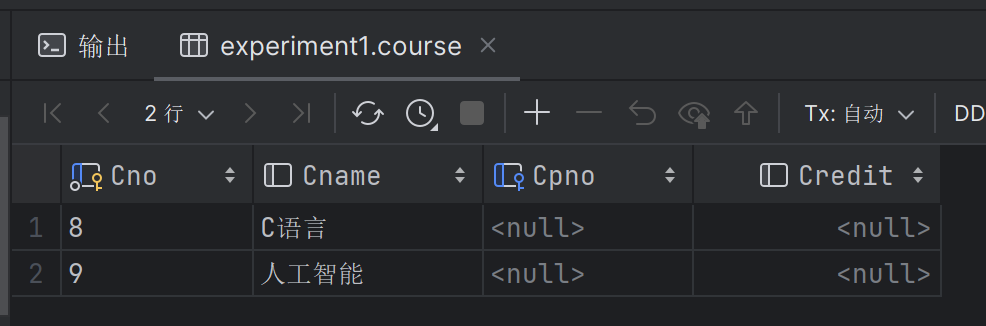


### 3.2.9（9）使用 insert 语句增加两门课程：C 语言和人工智能，并查询出来

代码：

insert into course(Cno,Cname) values (8,'C语言'),(9,'人工智能');  
select \* from course where Cname='C语言' or Cname='人工智能';

运行结果截图：

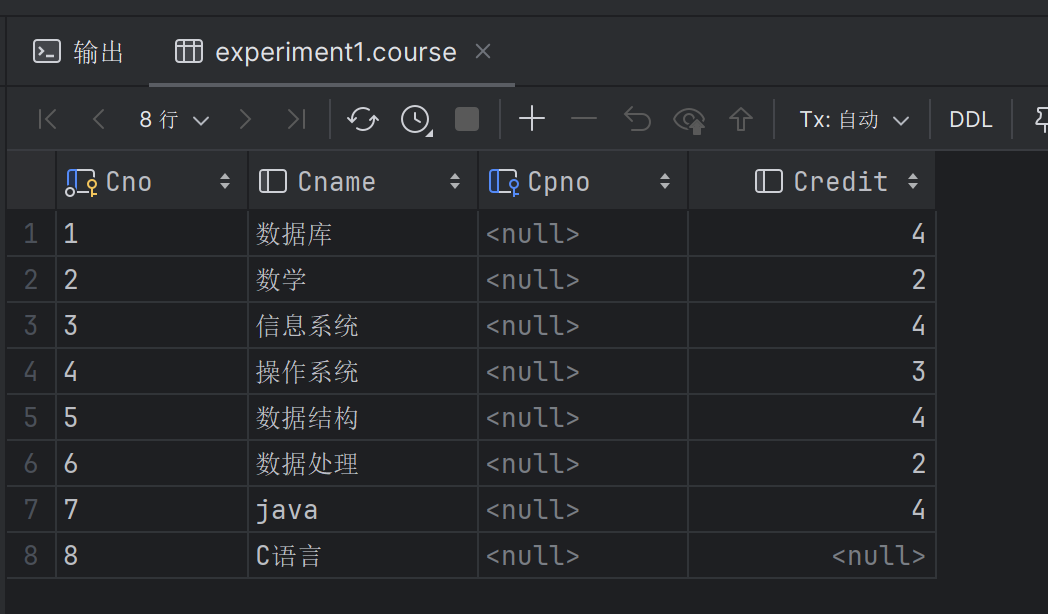


### 3.2.10（10）使用 delete 语句把人工智能课程删除，并查询出来。

代码：

delete from course where Cname='人工智能';  
select \* from course;

运行结果截图：



## 3.3任务总结

本次实验任务难度不大，通过实验指导书和查阅资料可以顺利完成，掌握SQL语言数据单表查询和多表查询，掌握了SQL语言基本的增删改查操作。

# 4 实验任务三（标题改为具体任务名称）

## 4.1 任务要求

(1)创建 CS 系的视图 CS\_View

(2)在视图 CS\_View 上查询 CS 系选修了 1 号课程的学生

(3)创建 IS 系成绩大于 80 的学生的视图 IS\_View

(4)在视图 IS\_View 查询 IS 系成绩大于 80 的学生

(5)删除视图 IS\_View

(6) 利用可视化窗口创建 2 个不同的用户 U1 和 U2,利用系统管理员给 U1 授予 Student 表的 查询和更新的权限，给 U2 对 SC 表授予插入的权限。然后用 U1 登录，分别 1）查询学生表 的信息；2）把所有学生的年龄增加 1 岁，然后查询；3）删除 IS 系的学生；4）查询 CS 系 的选课信息。用 U2 登录，分别 1）在 SC 表中插入 1 条记录（‘200215122’，‘1’，75）；2） 查询 SC 表的信息，3）查询视图 CS\_View 的信息。

(7) 用系统管理员登录，收回 U1 的所有权限

(8) 用 U1 登录，查询学生表的信息

(9) 用系统管理员登录

(10) 对 SC 表建立一个更新触发器，当更新了 SC 表的成绩时，如果更新后的成绩大于等于 95，则检查该成绩的学生是否有奖学金，如果奖学金是“否”，则修改为“是”。如果修改后的 成绩小于 95，则检查该学生的其他成绩是不是有大于 95 的，如果都没有，且修改前的成绩 是大于 95 时，则把其奖学金修改为”否”。然后进行成绩修改，并进行验证是否触发器正确 执行。1）首先把某个学生成绩修改为 98，查询其奖学金。2）再把刚才的成绩修改为 80， 再查询其奖学金。

（11）删除刚定义的触发器

（12）定义一个存储过程计算 CS 系的课程的平均成绩和最高成绩，在查询分析器或查询编 辑器中执行存储过程，查看结果。

（13）定义一个带学号为参数的查看某个学号的所有课程的成绩，查询结果要包含学生姓 名。进行验证。

（14）把上一题改成函数。再进行验证。

（15）在 SC 表上定义一个完整性约束，要求成绩再 0-100 之间。定义约束前，先把某个学 生的成绩修改成 120，进行查询，再修改回来。定义约束后，再把该学生成绩修改为 120， 然后进行查询。

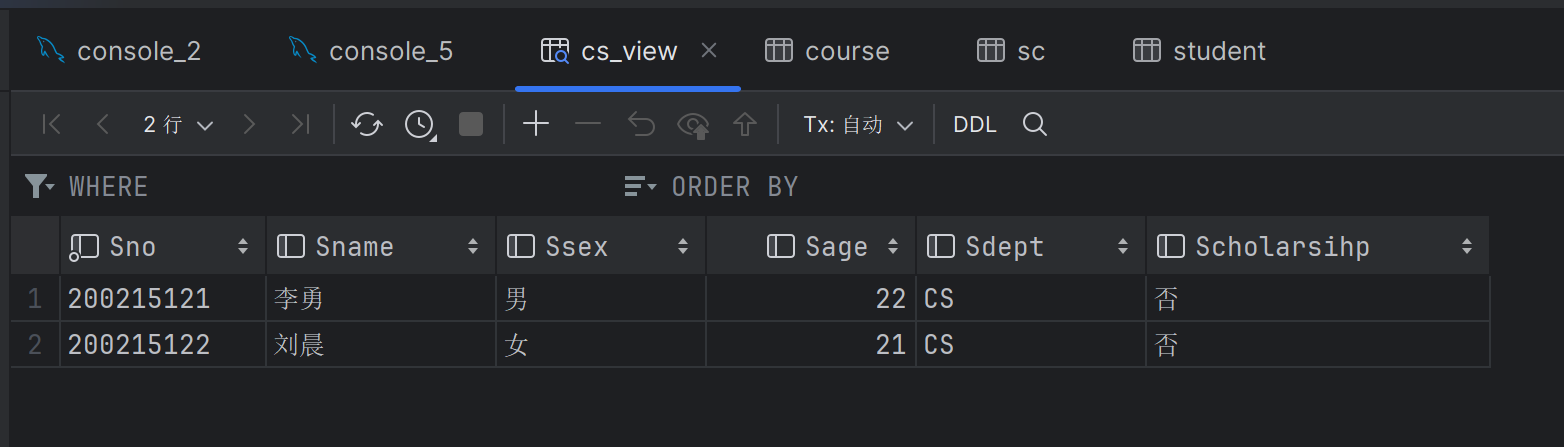
## 4.2 完成过程

### 4.2.1(1)创建 CS 系的视图 CS\_View

代码：

create view CS\_View as select \* from student where Sdept='CS';

运行结果截图：

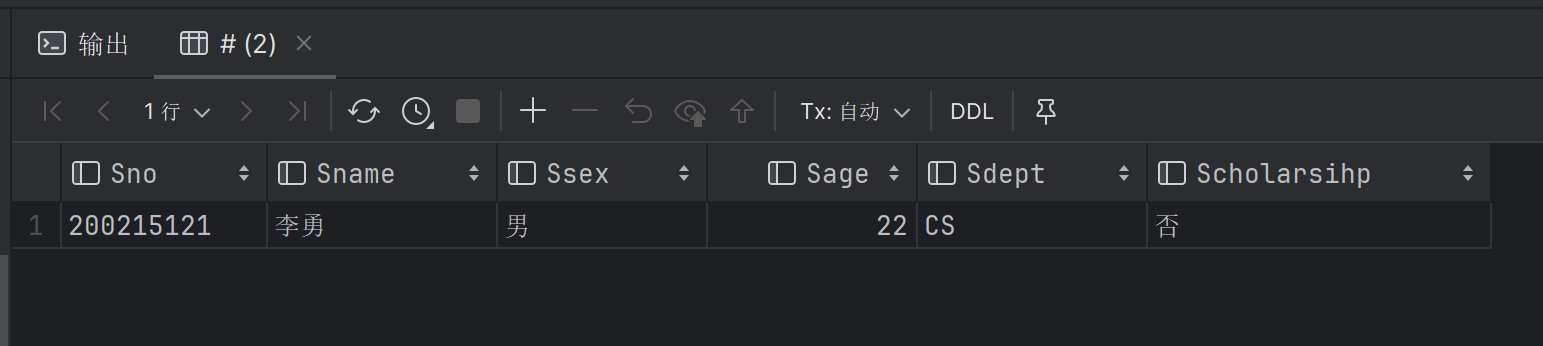


### 4.2.2(2)在视图 CS\_View 上查询 CS 系选修了 1 号课程的学生

代码：

select CS\_View.\* from CS\_View,sc where CS\_View.Sno=sc.Sno and sc.Cno='1';

运行结果截图：

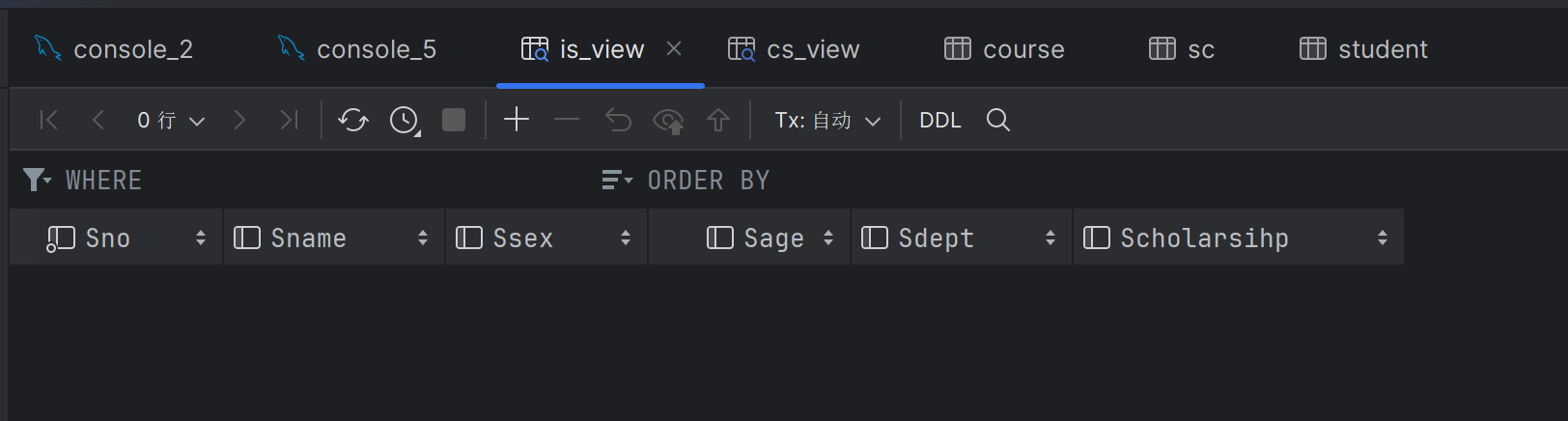


### 4.2.3(3)创建 IS 系成绩大于 80 的学生的视图 IS\_View

代码：

create view IS\_View as select student.\* from student,sc where student.Sno=sc.Sno and sc.Grade>80 and student.Sdept='IS';

运行结果截图：

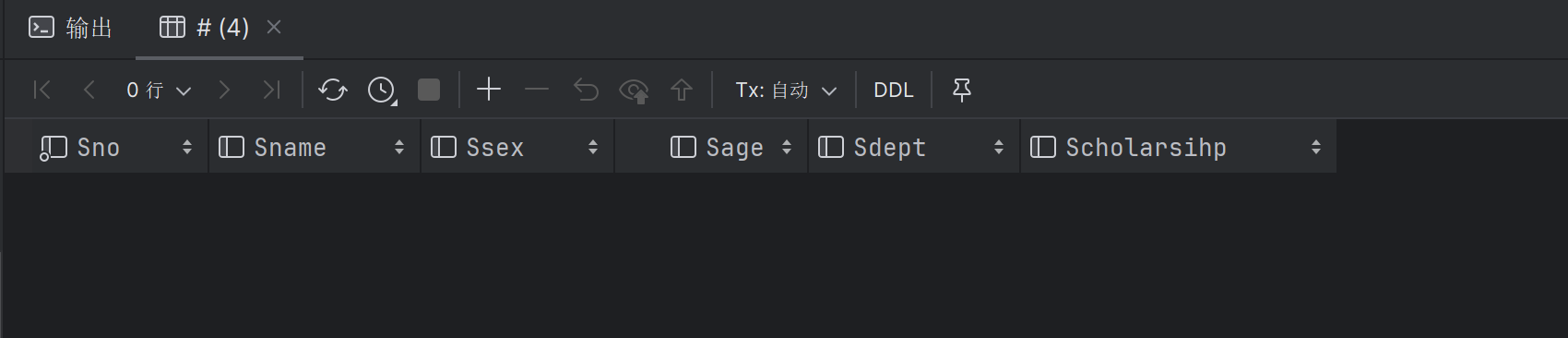


### 4.2.4(4)在视图 IS\_View 查询 IS 系成绩大于 80 的学生

代码:

select \* from IS\_View;

运行结果截图：

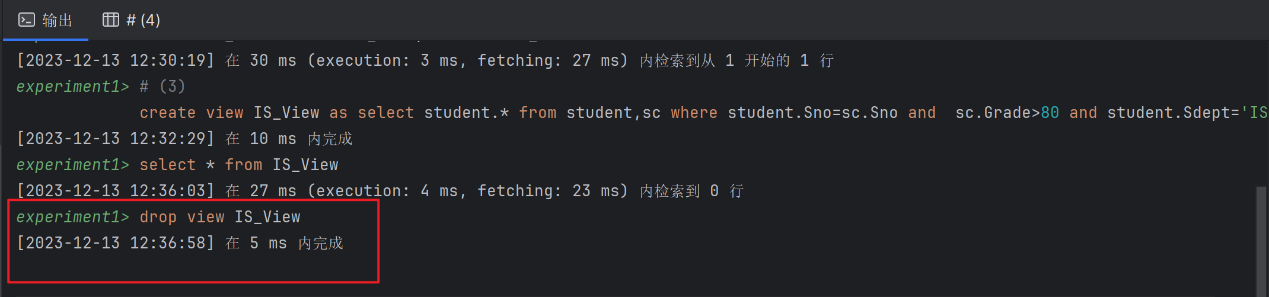


### 4.2.5(5)删除视图 IS\_View

代码：

drop view IS\_View;

运行结果截图：



### 4.2.6(6) 利用可视化窗口创建 2 个不同的用户 U1 和 U2,利用系统管理员给 U1 授予 Student 表的 查询和更新的权限，给 U2 对 SC 表授予插入的权限。然后用 U1 登录，分别 1）查询学生表 的信息；2）把所有学生的年龄增加 1 岁，然后查询；3）删除 IS 系的学生；4）查询 CS 系 的选课信息。用 U2 登录，分别 1）在 SC 表中插入 1 条记录（‘200215122’，‘1’，75）；2） 查询 SC 表的信息，3）查询视图 CS\_View 的信息。

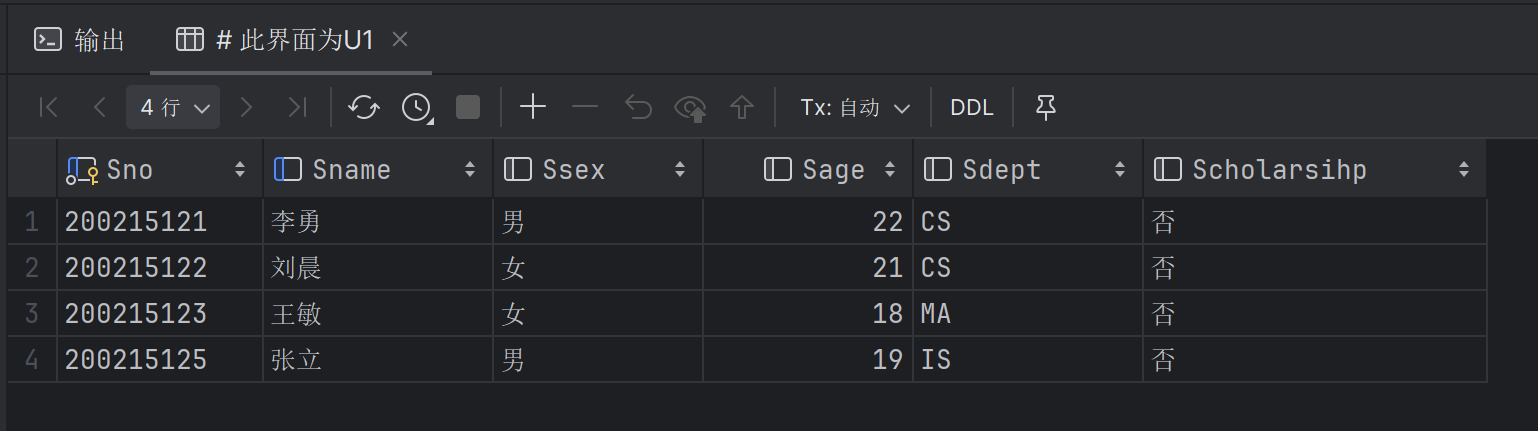
授权代码：

grant select,update on table student to U1;  
grant select, update on table experiment1.student to U1@localhost;  
grant insert on table experiment1.sc to U2@localhost;

U1代码及截图：

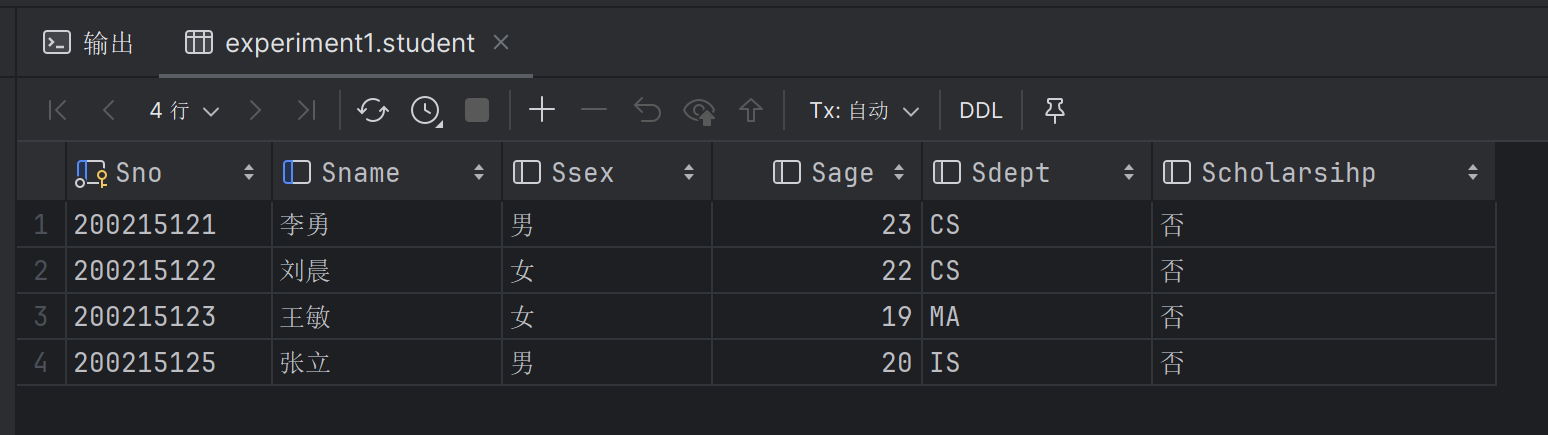
1）

select \*from student;



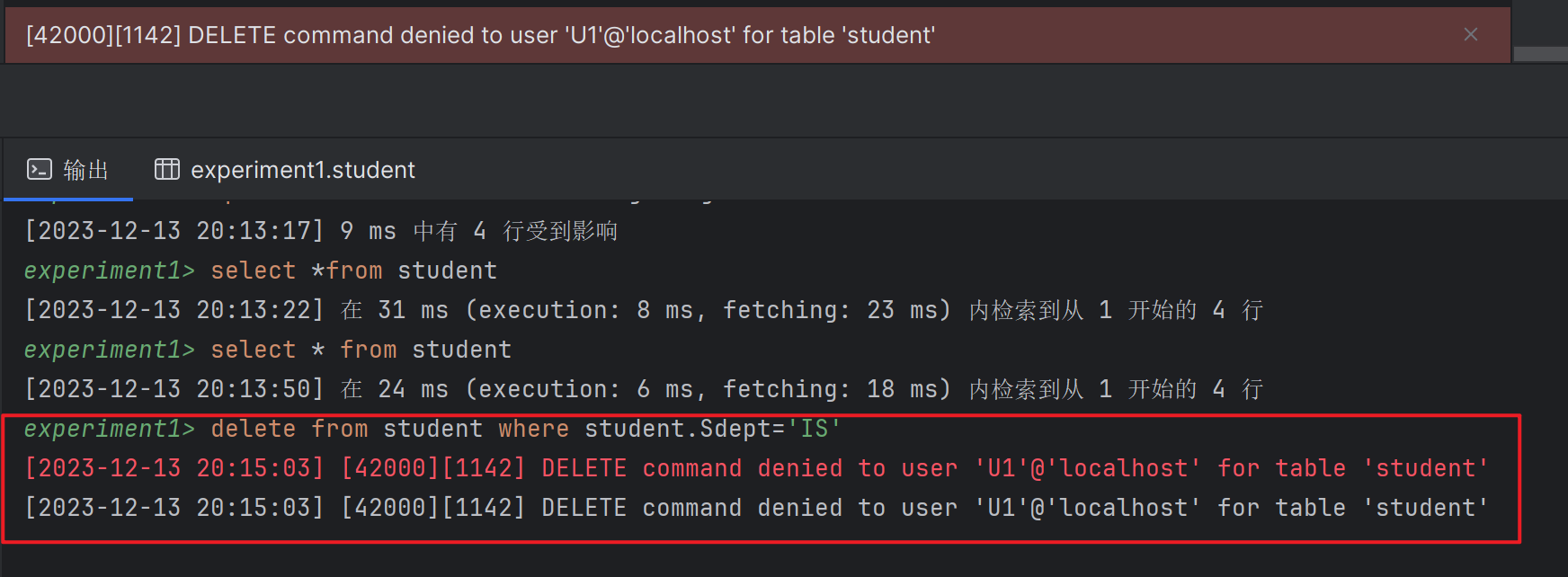
2）

update student set student.Sage=Sage+1;  
select \* from student;



3）

delete from student where student.Sdept='IS';



4）

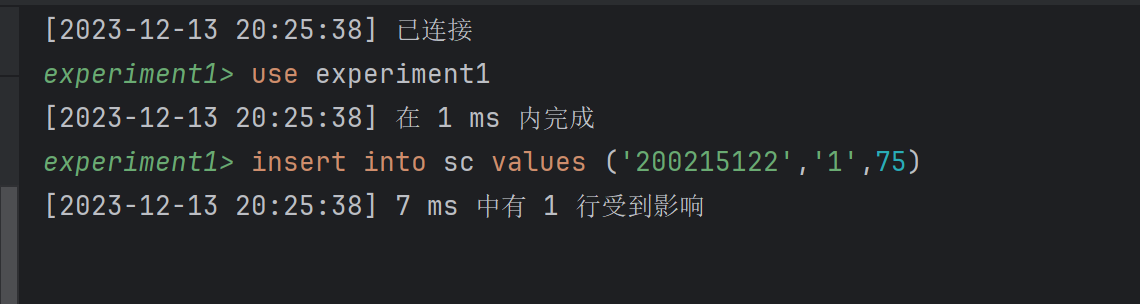
select \* from sc,student where sc.Sno=student.Sno and student.Sdept='CS';



U2代码及截图：

1）

insert into sc values ('200215122','1',75);



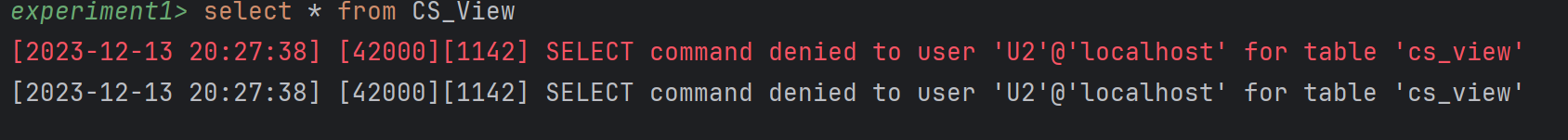
2）

select \* from sc;



3）

select \* from CS\_View;

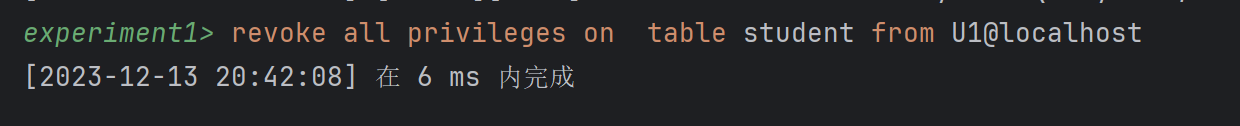


### 4.2.7(7) 用系统管理员登录，收回 U1 的所有权限

代码：

revoke all privileges on table student from U1@localhost;

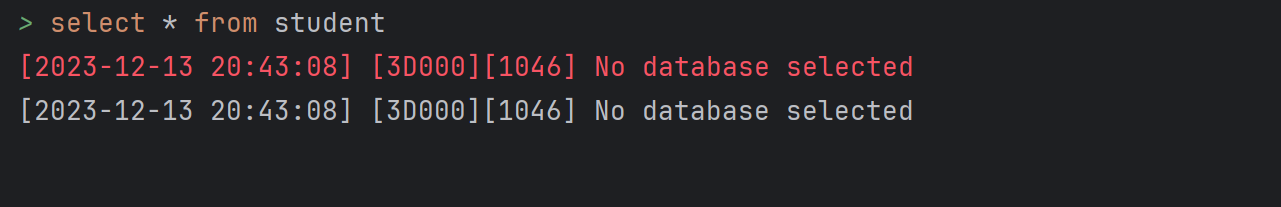
运行结果截图：



### 4.2.8(8) 用 U1 登录，查询学生表的信息

代码：

select \* from student;



### 3.2.10(10) 对 SC 表建立一个更新触发器，当更新了 SC 表的成绩时，如果更新后的成绩大于等于 95，则检查该成绩的学生是否有奖学金，如果奖学金是“否”，则修改为“是”。如果修改后的 成绩小于 95，则检查该学生的其他成绩是不是有大于 95 的，如果都没有，且修改前的成绩 是大于 95 时，则把其奖学金修改为”否”。然后进行成绩修改，并进行验证是否触发器正确 执行。1）首先把某个学生成绩修改为 98，查询其奖学金。2）再把刚才的成绩修改为 80， 再查询其奖学金。

代码：

create trigger test\_trigger after update on sc for each row  
 begin  
 update student set Scholarsihp='是' where (  
 select sc.sno, Grade,student.Scholarsihp,Sno from student,sc where sc.Sno=student.Sno and sc.Grade>95  
 );  
 end;

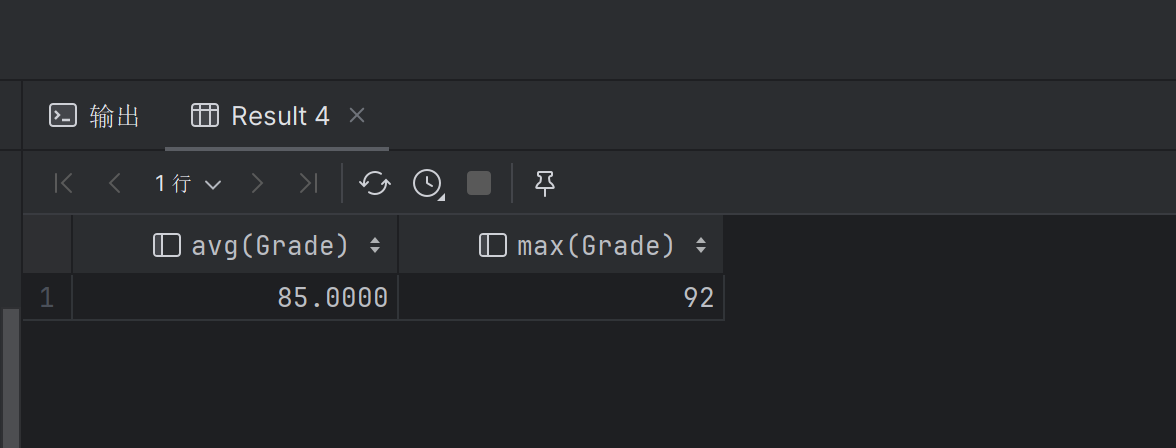
### 3.2.11（11）删除刚定义的触发器

### 3.2.12（12）定义一个存储过程计算 CS 系的课程的平均成绩和最高成绩，在查询分析器或查询编 辑器中执行存储过程，查看结果。

代码：

create procedure *p1*()  
begin  
 select *avg*(Grade),*max*(Grade) from sc, student where student.Sno=sc.Sno and student.Sdept='CS';  
end;  
call *p1*();

运行结果截图：

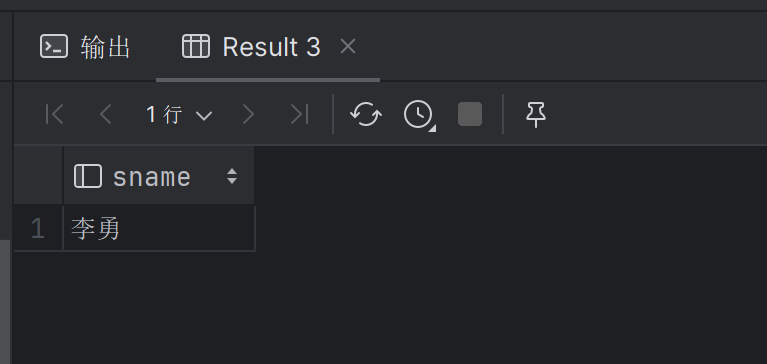


### 3.2.13（13）定义一个带学号为参数的查看某个学号的所有课程的成绩，查询结果要包含学生姓名。进行验证。

代码：

create procedure *p2*(in stu\_no varchar(9))  
begin  
 select student.sname from student where Sno=stu\_no;  
end;  
call *p2*('200215121');

运行结果截图：

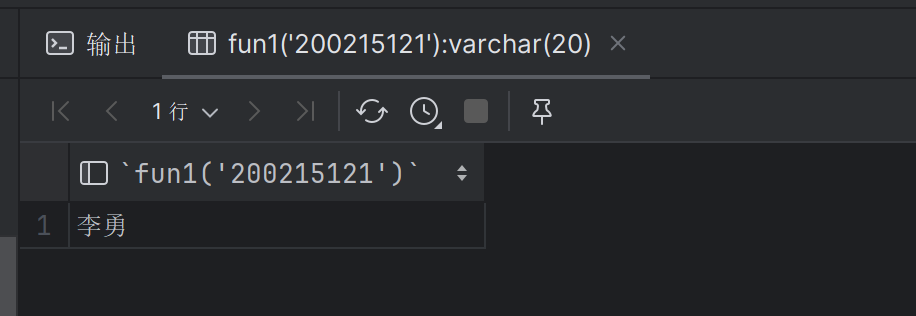


### 3.2.14（14）把上一题改成函数。再进行验证。

代码：

create function *fun1*(stu\_no varchar(9))  
returns varchar(20) deterministic  
begin  
 declare stu\_name varchar(20);  
 select student.sname into stu\_name from student where student.Sno=stu\_no;  
 return stu\_name;  
end;  
select *fun1*('200215121');

运行结果截图：

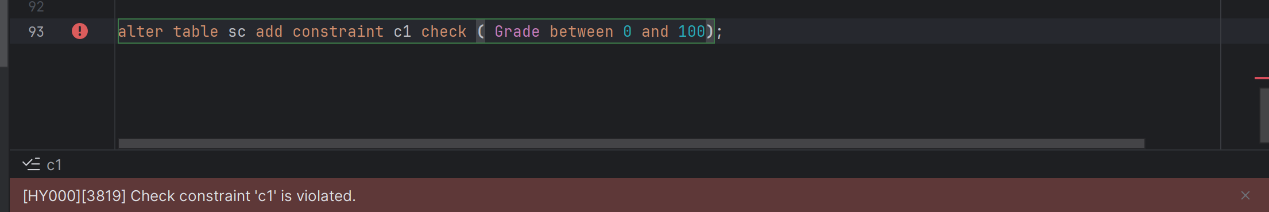


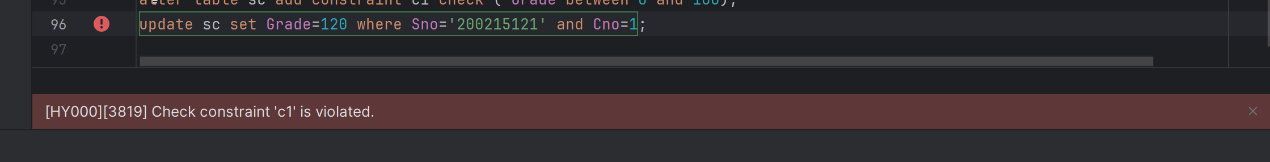
### 4.2.15（15）在 SC 表上定义一个完整性约束，要求成绩再 0-100 之间。定义约束前，先把某个学 生的成绩修改成 120，进行查询，再修改回来。定义约束后，再把该学生成绩修改为 120， 然后进行查询。

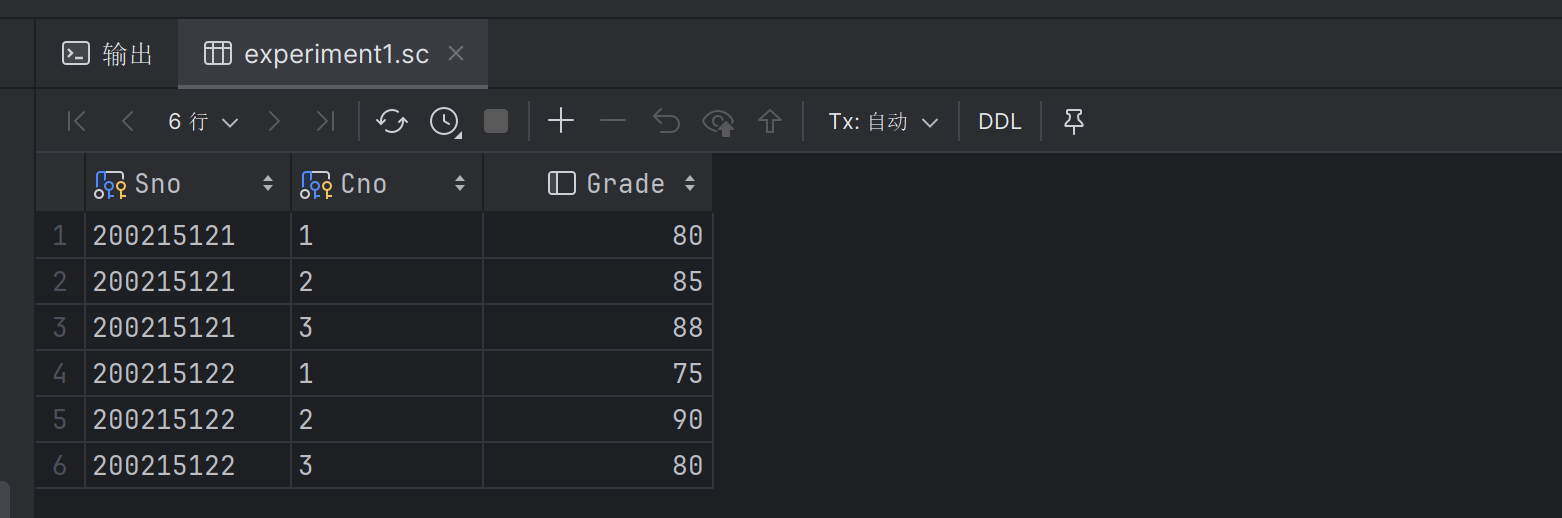
代码：

alter table sc add constraint c1 check ( Grade between 0 and 100);  
update sc set Grade=80 where Sno='200215121' and Cno=1;  
alter table sc add constraint c1 check ( Grade between 0 and 100);  
update sc set Grade=120 where Sno='200215121' and Cno=1;  
select \* from sc;

运行结果截图：







## 4.3任务总结

本次实验中熟悉了视图，触发器，存储过程，函数的基本概念和用法，掌握了表的试图的创建，查询和删除，掌握了存储过程的相关操作，学习了如何使用函数封装SQL语句。

# 5 实验任务四（标题改为具体任务名称）

## 5.1 任务要求

熟练掌握使用 SQL 语句设计数据库的方法，实现前述实验的学生管理系统，完成实验报告

1）新生入学信息增加，学生信息修改。

2）课程信息维护（增加新课程，修改课程信息，删除没有选课的课程信息）。

3）录入学生成绩，修改学生成绩。

4）按系统计学生的平均成绩、最好成绩、最差成绩、优秀率、不及格人数。

5）按系对学生成绩进行排名，同时显示出学生、课程和成绩信息。

6）输入学号，显示该学生的基本信息和选课信息。

## 5.2 完成过程

本实验采用前后端分离的设计，前端使用HTML,CSS,JavaScript实现界面，发送网络请求，后端在node.js环境下使用express后端框架，处理前端请求，连接数据库。

项目目录结构

student\_manage

│ package-lock.json//node配置文件

│ package.json

│

├─.vscode

│ settings.json//vscode配置文件

│

├─be//后端代码文件夹

│ data.sql//初始化数据库的文件

│ server.js//服务器文件

│ sql.js//封装的express连接数据库的配置文件

│

├─fe//前端代码文件夹

│ ├─pages//前端html文件夹

│ │ course.html//课程信息页面

│ │ login.html//登录页面

│ │ personalGrade.html//个人成绩查询页面

│ │ sc.html//选课信息页面

│ │ statistics.html//各个院系数据统计页面

│ │ student.html//学生信息页面

│ │

│ ├─scripts//前端JavaScript文件夹

│ │ commo.js//公共的js文件，用于顶部导航栏的路由导航

│ │ course.js//course页面所需的js文件包括dom操作和网络请求，下

│ │ 同

│ │ login.js

│ │ personalGrade.js

│ │ sc.js

│ │ statistics.js

│ │ student.js

│ │

│ └─style//前端css文件夹

│ common.css//公共的样式文件

│ login.css//登录页面的样式文件

│ personalGrade.css//个人信息查询的样式文件

│

└─node\_modules//npm管理的模块

### （1）实现步骤：

后端：

**1） 环境准备：**

本次实验后端实验环境在node环境下进行，使用npm包管理工具，安装依赖如下：

npm i

npm i cors

npm i express

npm i mysql

各个模块作用：

cors：解决前后端跨域问题

express：node.js后端所用框架

mysql：node.js连接数据库所依赖的模块

**2） 后端代码：**

后端核心代码：

//data.sql

-- 这个项目的前置数据库初始化

create table Student(

    Sno char(9) primary key,

    Sname char(20) unique,

    Ssex char(2),

    Sage smallint,

    Sdept char(20),

    Scholarship char(2)

);

create table Course(

    Cno char(4) primary key,

    Cname char(40),

    Cpno char(4),

    Credit smallint,

    foreign key (Cpno) references Course(Cno)

);

create table SC(

    Sno char(9),

    Cno char(4),

    Grade smallint,

    primary key (Sno,Cno),

    foreign key (Sno) references Student(Sno),

    foreign key (Cno) references Course(Cno)

);

create table user(

    id int primary key auto\_increment,

    username varchar(20) unique ,

    password varchar(20)

    );

insert into student values

('200215121','李勇','男',20,'CS','否'),

('200215122','刘晨','女',19,'CS','否'),

('200215123','王敏','女',18,'MA','否'),

('200215125','张立','男',19,'IS','否');

insert into course values

('1','数据库',NULL,4),

('2','数学',NULL,2),

('3','信息系统',NULL,4),

('4','操作系统',NULL,3),

('5','数据结构',NULL,4),

('6','数据处理',NULL,2),

('7','java',NULL,4);

insert into SC values

('200215121','1',92),

('200215121','2',85),

('200215121','3',88),

('200215122','2',90),

('200215122','3',80);

insert into user value (1,"bowl",123456);

//sever.js

const express = require('express');

const db = require('./sql')//配置数据库

const cors = require('cors');//解决跨域

const server = express();

server.use(cors());//解决跨域问题的插件

//登录api

server.get('/api/login', (req, res) => {

    let username = req.query.username;

    let password = req.query.password;

    db.query('select \* from user where username=? and password=?', [username, password], (err, data) => {

        if (err) {

            throw err;

        }

        else if (data.length > 0) {

            res.end(JSON.stringify(data));

        }

    })

})

//student界面的api

//新增学生

server.get('/api/student/add', (req, res) => {

    console.log(req.query);

    let Sno = req.query.Sno;

    let Sname = req.query.Sname;

    let Ssex = req.query.Ssex;

    let Sage = req.query.Sage;

    let Sdept = req.query.Sdept;

    let Scholarship = req.query.Scholarship;

    let flag = Sno && Sname && Ssex && Sage && Sdept && Scholarship

    console.log(flag);

    if (flag) {

        db.query("insert into student values (?,?,?,?,?,?)", [Sno, Sname, Ssex, Sage, Sdept, Scholarship], (err, data) => {

            if (err)

                throw err;

        })

        res.end("新增学生信息成功");

    }

    else {

        res.end('请正确填写完整学生信息')

    }

})

//查询学生

server.get('/api/student/select', (req, res) => {

    db.query('select \* from student', (err, data) => {

        if (err) {

            throw err;

        }

        else {

            res.end(JSON.stringify(data));

        }

    })

})

//更新学生，根据学号修改，其他修改暂时懒得搞

server.get('/api/student/update', (req, res) => {

    console.log('update');

    let Sno = req.query.Sno;

    let Sname = req.query.Sname;

    let Ssex = req.query.Ssex;

    let Sage = req.query.Sage;

    let Sdept = req.query.Sdept;

    let Scholarship = req.query.Scholarship;

    let flag = Sno && Sname && Ssex && Sage && Sdept && Scholarship

    console.log(Sno, Sname, Ssex, Sage, Sdept, Scholarship);

    if (flag) {

        db.query("update student set Sname=?, Ssex=? ,Sage=?, Sdept=?, Scholarship=? where student.Sno=?;", [Sname, Ssex, Sage, Sdept, Scholarship, Sno], (err, data) => {

            if (err)

                throw err;

            console.log(data);

        })

        res.end("更改学生信息成功");

    }

    else {

        res.end('请正确填写完整学生信息')

    }

}

)

//删除学生

server.get('/api/student/delete', (req, res) => {

    let Sno = req.query.Sno;

    let Sname = req.query.Sname;

    let Ssex = req.query.Ssex;

    let Sage = req.query.Sage;

    let Sdept = req.query.Sdept;

    let Scholarship = req.query.Scholarship;

    let flag = Sno && Sname && Ssex && Sage && Sdept && Scholarship

    console.log(Sno, Sname, Ssex, Sage, Sdept, Scholarship);

    if (flag) {

        db.query("delete from student where Sno=?;", [Sno], (err, data) => {

            if (err)

                throw err;

            console.log(data);

        })

        res.end("删除学生信息成功");

    }

    else {

        res.end('请正确填写完整学生信息')

    }

})

// sc界面的api

//选课查询

server.get('/api/sc/select', (req, res) => {

    db.query('select \* from sc', (err, data) => {

        if (err) {

            throw err;

        }

        else {

            res.end(JSON.stringify(data));

        }

    })

})

// 选课增加

server.get('/api/sc/add', (req, res) => {

    console.log(req.query);

    let Sno = req.query.Sno;

    let Cno = req.query.Cno;

    let Grade = req.query.Grade;

    let flag = Sno && Cno && Grade;

    if (flag) {

        db.query('insert into sc values (?,?,?)', [Sno, Cno, Grade], (err, data) => {

            if (err)

                throw err;

        })

        res.end('新增选课成功')

    }

    else {

        res.end('请填写正确选课信息');

    }

})

// 选课修改

server.get('/api/sc/update', (req, res) => {

    console.log(req.query);

    let Sno = req.query.Sno;

    let Cno = req.query.Cno;

    let Grade = req.query.Grade;

    let flag = Sno && Cno && Grade;

    if (flag) {

        db.query('update sc set Grade=? where Sno=? and Cno=?', [Grade, Sno, Cno], (err, data) => {

            if (err)

                throw err;

        })

        res.end('更改选课成功')

    }

    else {

        res.end('请填写正确选课信息');

    }

})

// 选课删除

server.get('/api/sc/delete', (req, res) => {

    let Sno = req.query.Sno;

    let Cno = req.query.Cno;

    let Grade = req.query.Grade;

    let flag = Sno && Cno && Grade;

    if (flag) {

        db.query('delete from sc where Sno=? and Cno=?;', [Sno, Cno], (err, data) => {

            if (err)

                throw err;

        })

        res.end('删除选课信息成功')

    }

    else {

        res.end('请正确填写完整选课信息');

    }

})

// course界面的api

//课程查询

server.get('/api/course/select', (req, res) => {

    db.query('select \* from course', (err, data) => {

        if (err) {

            throw err;

        }

        else {

            res.end(JSON.stringify(data));

        }

    })

})

// 课程新增

server.get('/api/course/add', (req, res) => {

    console.log(req.query);

    let Cno = req.query.Cno;

    let Cname = req.query.Cname;

    let Cpno = req.query.Cpno;

    let Credit = req.query.Credit;

    let flag = Cno && Cname && Cpno && Credit;

    if (flag) {

        db.query('insert into course values (?,?,?,?)', [Cno, Cname, Cpno, Credit], (err, data) => {

            if (err)

                throw err;

        })

        res.end('新增课程成功')

    }

    else {

        res.end('请填写正确课程信息');

    }

})

// 课程修改

server.get('/api/course/update', (req, res) => {

    console.log(req.query);

    let Cno = req.query.Cno;

    let Cname = req.query.Cname;

    let Cpno = req.query.Cpno;

    let Credit = req.query.Credit;

    let flag = Cno && Cname && Cpno && Credit;

    if (flag) {

        db.query('update course set Cname=?, Cpno=?, Credit=? where Cno=?', [Cname, Cpno, Credit, Cno], (err, data) => {

            if (err)

                throw err;

        })

        res.end('更改课程成功')

    }

    else {

        res.end('请填写正确课程信息');

    }

})

// 课程删除

server.get('/api/course/delete', (req, res) => {

    let Cno = req.query.Cno;

    let Cname = req.query.Cname;

    let Cpno = req.query.Cpno;

    let Credit = req.query.Credit;

    let flag = Cno && Cname && Cpno && Credit;

    if (flag) {

        db.query('delete from course where Cno=? and Cname=?;', [Cno, Cname], (err, data) => {

            if (err)

                throw err;

        })

        res.end('删除课程信息成功')

    }

    else {

        res.end('请正确填写完整课程信息');

    }

})

// cs学生成绩查询

server.get('/api/statistics/cs', (req, res) => {

    db.query('select student.Sdept sdeption, avg(Grade) avg,max(Grade) max,min(Grade) min from sc,student where Student.Sno=sc.Sno GROUP BY Sdept;', (err, data) => {

        if (err) {

            throw err;

        }

        else {

            res.end(JSON.stringify(data))

        }

    })

})

// 个人信息查询

server.get('/api/personalGrade', (req, res) => {

    let Sno = req.query.Sno;

    db.query('select \* from student where student.Sno=?', [Sno], (err, data) => {

        if (err) {

            throw err;

        }

        else {

            res.end(JSON.stringify(data));

        }

    })

})

//监听3000端口

server.listen(3000, () => {

    console.log('服务已启动');

});

//sql.js

var mysql = require('mysql');

var db = mysql.createConnection({

    host: 'localhost',

    user: 'root',

    password: '123456',

    database: 'student\_manage'

})

db.connect();

module.exports = db;

**3）后端代码使用方法：**

在cmd中输入node server.js即可开启服务器代码，服务器监听3000端口的网络请求。

前端：

1. 环境准备：

前端代码使用原生HTML，CSS，JavaScript实现，网络请求使用axios模块，导入方式为cdn导入。

1. 前端代码：

登录界面：

//login.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="../style/login.css">

    <title>登录</title>

</head>

<body>

    <form  class="container">

        <input type="text" name="username" id="username">

        <label class="usernameLabel" placeholder=" ">用户名</label>

        <input type="password" name="password" id="password">

        <label class="passwordLabel" placeholder=" ">密码</label>

        <div class="loginBtn">登录</div>

    </form>

    <div class="title">

        <div class="first">Manage</div>

        <div class="second">Your</div>

        <div class="third">Student</div>

        <div class="forth">Information</div>

        <div class="fifth">Effectively</div>

    </div>

</body>

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<script src="../scripts/login.js"></script>

</html>

//login.css

\* {

    margin: 0;

    padding: 0;

}

body {

    width: 100vw;

    height: 100vh;

    overflow: hidden;

    background-color: #1E1E1E;

}

.container {

    width: 600px;

    height: 700px;

    margin: 100px;

    float: left;

    display: flex;

    flex-direction: column;

    justify-content: center;

    align-items: center;

    background-color: #2d2d2d;

    border-radius: 50px;

    position: relative;

}

#username,

#password {

    width: 300px;

    height: 30px;

    border: none;

    border-bottom: 1px solid #454545;

    background-color: transparent;

    margin-bottom: 100px;

    font-size: 20px;

    color: #006fff;

    user-select: none;

    /\* pointer-events: none; \*/

}

input:focus {

    outline: none;

}

label {

    position: absolute;

    color: #9e9e9e;

    font-size: 25px;

    left: 150px;

    transition: 0.2s ease all;

    user-select: none;

}

.usernameLabel {

    top: 180px;

}

.passwordLabel {

    top: 310px

}

input:focus+label {

    transform: translateY(-30px);

    font-size: 20px;

    color: #006fff;

}

.loginBtn {

    width: 100px;

    height: 50px;

    margin-top: 20px;

    display: flex;

    justify-content: center;

    align-items: center;

    border-radius: 20px;

    cursor: pointer;

    font-size: 30px;

    color: #9e9e9e;

    user-select: none;

}

.loginBtn:hover {

    color: #006fff;

}

.title {

    float: left;

    /\* background-color: pink; \*/

    font-size: 100px;

    margin: 100px 0 0 0;

    user-select: none;

}

.first {

    color: #6c35de;

}

.second {

    color: #3F51B5;

}

.third {

    color: #006fff;

}

.forth {

    color: #0085ff;

}

.fifth {

    color: #00E5FF;

}

//login.js

document.querySelector('.loginBtn').addEventListener('click', () => {

    let username = document.querySelector('#username').value;

    let password = document.querySelector('#password').value;

    console.log(username);

    console.log('click');

    axios({

        url: 'http://localhost:3000/api/login',

        params: {

            username: username,

            password: password

        }

    }).then(result => {

        let correctUsername = result.data[0].username;

        let correctPassword = result.data[0].password;

        if (correctPassword === password && correctUsername === username) {

            alert('登录成功');

            window.location.href = 'student.html';

        }

        else {

            alert('登录失败');

        }

    })

});

//主要功能页面（以学生信息查询页面为例，后续页面代码见实验源码）：

//student.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="../style/common.css">

    <!-- <link rel="stylesheet" href="../style/student.css"> -->

    <title>学生信息</title>

</head>

<body>

    <div class="top">

        <div class="student">学生信息</div>

        <div class="sc">选课信息</div>

        <div class="course">课程信息</div>

        <div class="statistics">成绩统计</div>

        <div class="personalGrade">个人成绩</div>

    </div>

     <div class="dataFunction">

        <div class="selectBtn">查询</div>

        <div class="addBtn">增加</div>

        <div class="updateBtn">修改</div>

        <div class="deleteBtn">删除</div>

    </div>

    <form class="center">

        <label>学号</label>

        <input type="text" id="Sno" name="Sno">

        <label >姓名</label>

        <input type="text" id="Sname" name="Sname">

        <label >性别</label>

        <input type="text"  id="Ssex" name="Ssex">

        <label >年龄</label>

        <input type="text" id="Sage" name="Sage">

        <label >系别</label>

        <input type="text" id="Sdept" name="Sdept">

        <label >奖学金</label>

        <input type="text" id="Scholarship" name="Scholarship">

    </form>

    <table class="dataList">

        <tr class="tableHead"></tr>

        <tbody class="tableBody"></tbody>

    </table>

</body>

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<script src="../scripts/student.js"></script>

<script src="../scripts/commo.js"></script>

</html>

//common.css

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    width: 100vw;

    background-color: #454545;

    user-select: none;

    color: #FFFFFF;

}

.top {

    width: 100%;

    height: 40px;

    float: left;

    background-color: #2d2d2d;

    display: flex;

    justify-content: space-evenly;

}

.top>div {

    height: 40px;

    line-height: 40px;

}

.top>div:hover {

    border-bottom: 2px solid #006fff;

}

.dataList,

th,

td {

    border: solid;

    border-collapse: collapse;

    vertical-align: center;

    text-align: center;

}

th,

td {

    height: 100px;

}

.student,

.sc,

.course,

.statistics,

.personalGrade {

    cursor: pointer;

}

.dataList {

    margin-top: 20px;

    width: 100vw;

}

.dataFunction {

    width: 100vw;

    height: 40px;

    display: flex;

    justify-content: space-evenly;

    background-color: #2d2d2d;

    margin-bottom: 10px;

}

.dataFunction>div {

    cursor: pointer;

    height: 40px;

    line-height: 40px;

}

.dataFunction>div:hover {

    border-bottom: 2px solid #006fff;

}

//student.js

const tableHead = document.querySelector('.tableHead');

const tableBody = document.querySelector('.tableBody');

// 获取四个功能按钮

const addBtn = document.querySelector('.addBtn');

const updateBtn = document.querySelector('.updateBtn');

const selectBtn = document.querySelector('.selectBtn');

const deleteBtn = document.querySelector('.deleteBtn');

// 获取输入框

let Sno = document.querySelector('#Sno');

let Sname = document.querySelector('#Sname');

let Ssex = document.querySelector('#Ssex');

let Sage = document.querySelector('#Sage');

let Sdept = document.querySelector('#Sdept');

let Scholarship = document.querySelector('#Scholarship');

// 增

addBtn.addEventListener('click', () => {

    console.log('click');

    console.log(Sno.value, Sname.value, Ssex.value, Sage.value, Sdept.value, Scholarship.value);

    axios({

        url: 'http://localhost:3000/api/student/add',

        params: {

            Sno: Sno.value,

            Sname: Sname.value,

            Ssex: Ssex.value,

            Sage: Sage.value,

            Sdept: Sdept.value,

            Scholarship: Scholarship.value

        }

    }).then(result => {

        alert(result.data);

    })

    window.location.reload();

})

// 改

updateBtn.addEventListener('click', () => {

    console.log('update');

    axios({

        url: 'http://localhost:3000/api/student/update',

        params: {

            Sno: Sno.value,

            Sname: Sname.value,

            Ssex: Ssex.value,

            Sage: Sage.value,

            Sdept: Sdept.value,

            Scholarship: Scholarship.value

        }

    }).then(result => {

        alert(result.data);

    })

    window.location.reload();

})

// 查

selectBtn.addEventListener('click', () => {

    axios({

        url: 'http://localhost:3000/api/student/select'

    }).then(result => {

        let head = ['学号', '姓名', '性别', '年龄', '系别', '奖学金'].map(item => `<th>${item}</th>`)

        let newlist = result.data.map(item =>

            `<tr class='listItem'>

            <td>${item.Sno}</td>

            <td>${item.Sname}</td>

            <td>${item.Ssex}</td>

            <td>${item.Sage}</td>

            <td>${item.Sdept}</td>

            <td>${item.Scholarship}</td>

            </tr>`

        )

        tableHead.innerHTML = head.join('');

        tableBody.innerHTML = newlist.join('');

    })

})

// 删

deleteBtn.addEventListener('click', () => {

    axios({

        url: 'http://localhost:3000/api/student/delete',

        params: {

            Sno: Sno.value,

            Sname: Sname.value,

            Ssex: Ssex.value,

            Sage: Sage.value,

            Sdept: Sdept.value,

            Scholarship: Scholarship.value

        }

    }).then(result => {

        alert(result.data);

    })

    window.location.reload();

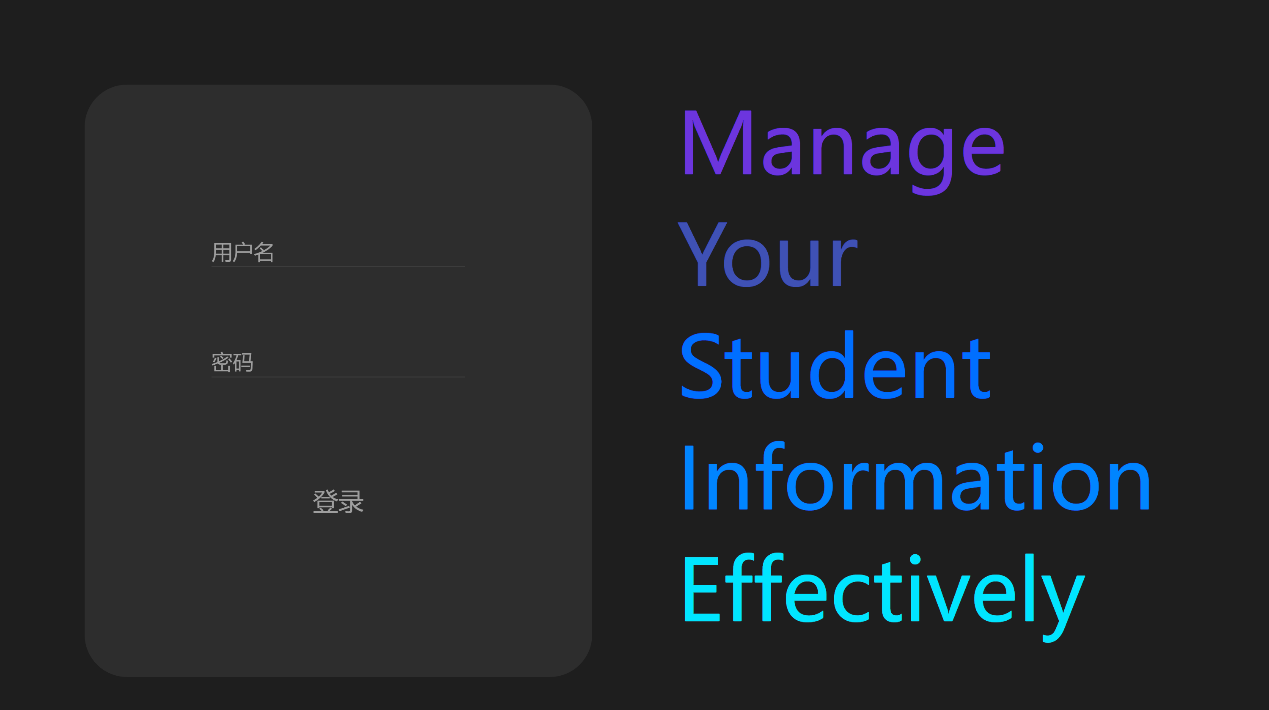
})

1. 前端代码使用方法：

入口文件为login.html，双击打开即可，在chrome浏览器下运行效果最佳。

项目具体使用及实验要求实现：

1. 登录：

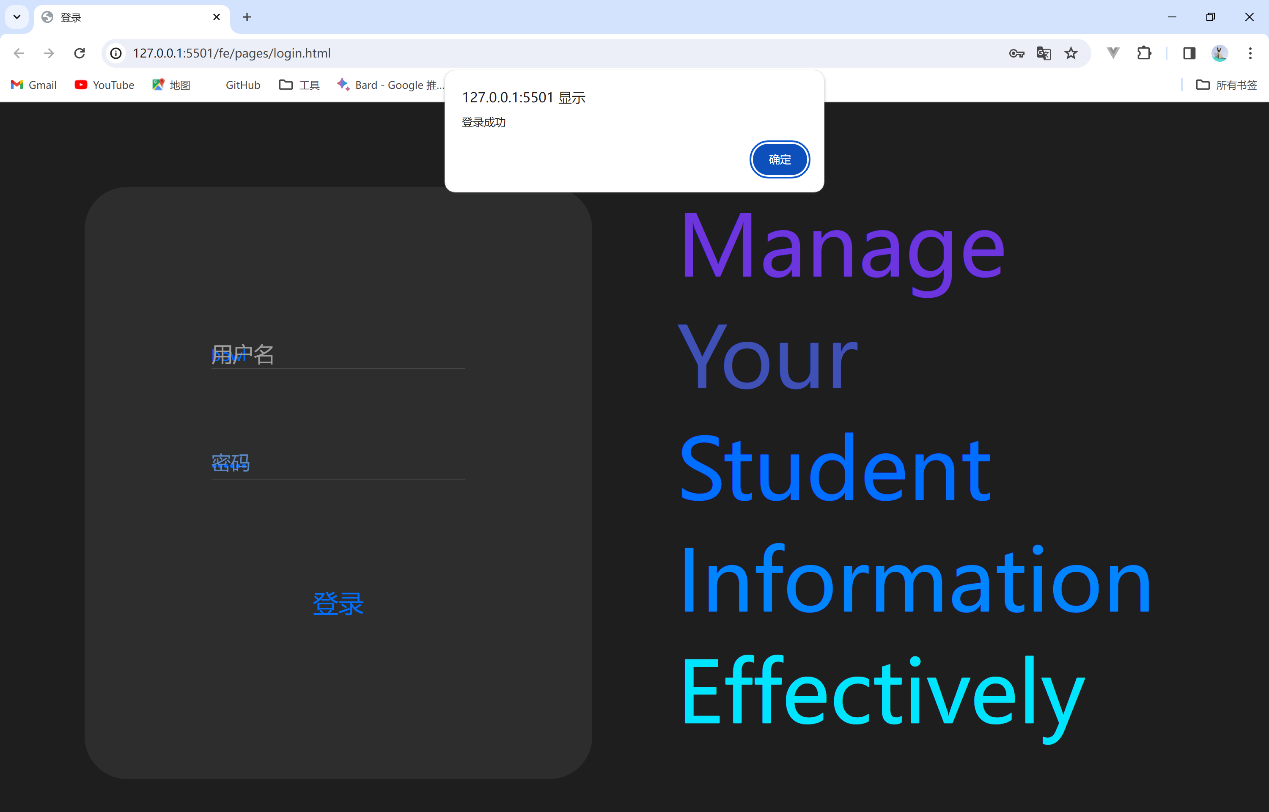


用户名：bowl

密码：123456

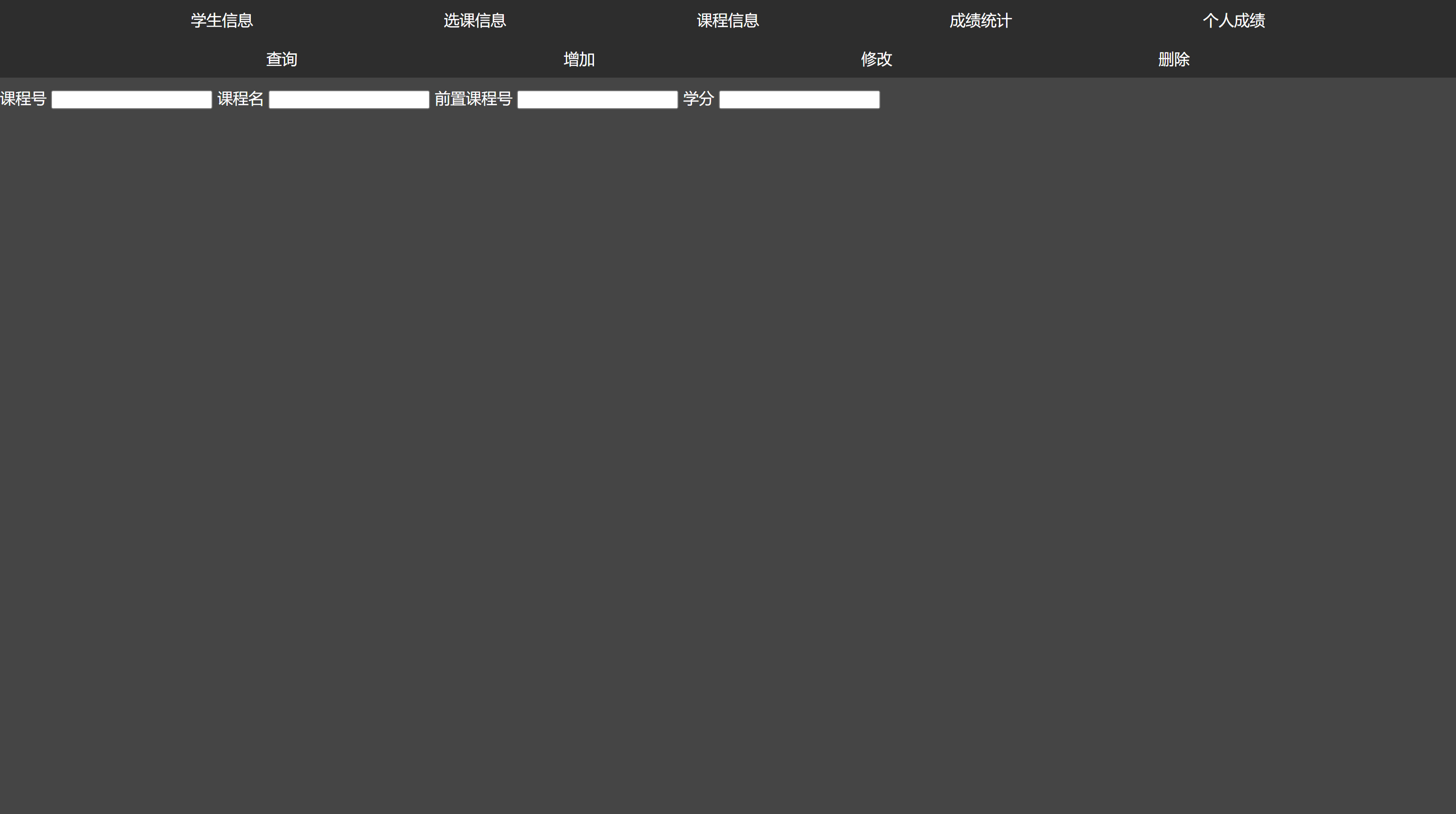
若需添加用户，在数据库中添加新的用户名及密码即可。

登录成功弹窗：



1. 具体功能页面

登录成功后默认进入学生信息界面



顶部导航栏可切换页面，各个页面的功能如下

学生信息：

可对student表进行增删改查，从而操作学生信息。

选课信息：

可对sc表进行增删改查，从而操作选课和成绩信息。

课程信息：

可对course表进行增删改查，从而操作课程信息。

成绩统计：

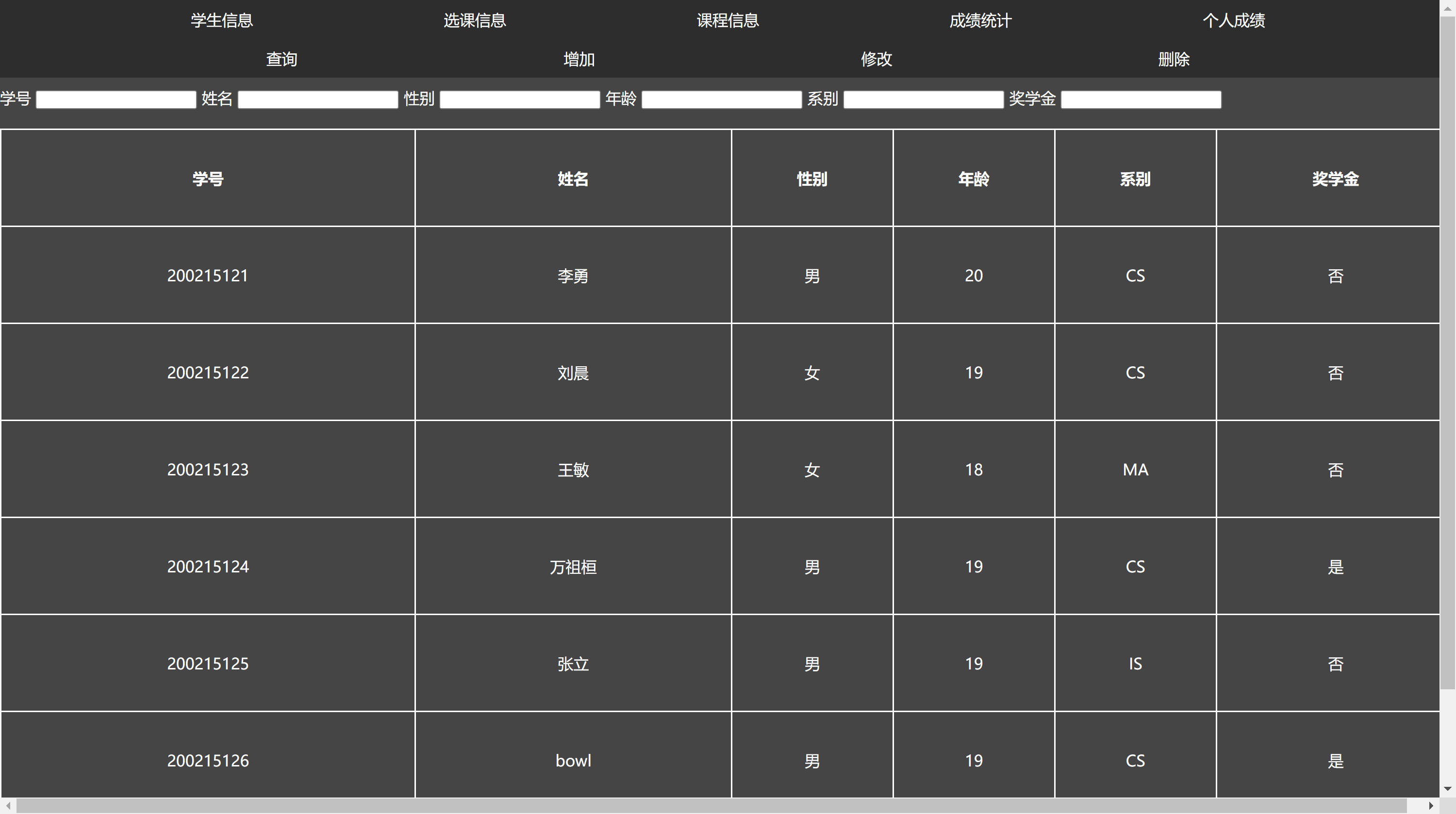
统计各个院系的总体成绩，包括最高分，最低分，平均成绩，优秀率。

个人成绩：

输入学号即可查询学生成绩。

以学生信息页面为例（选课信息和课程信息同理）：

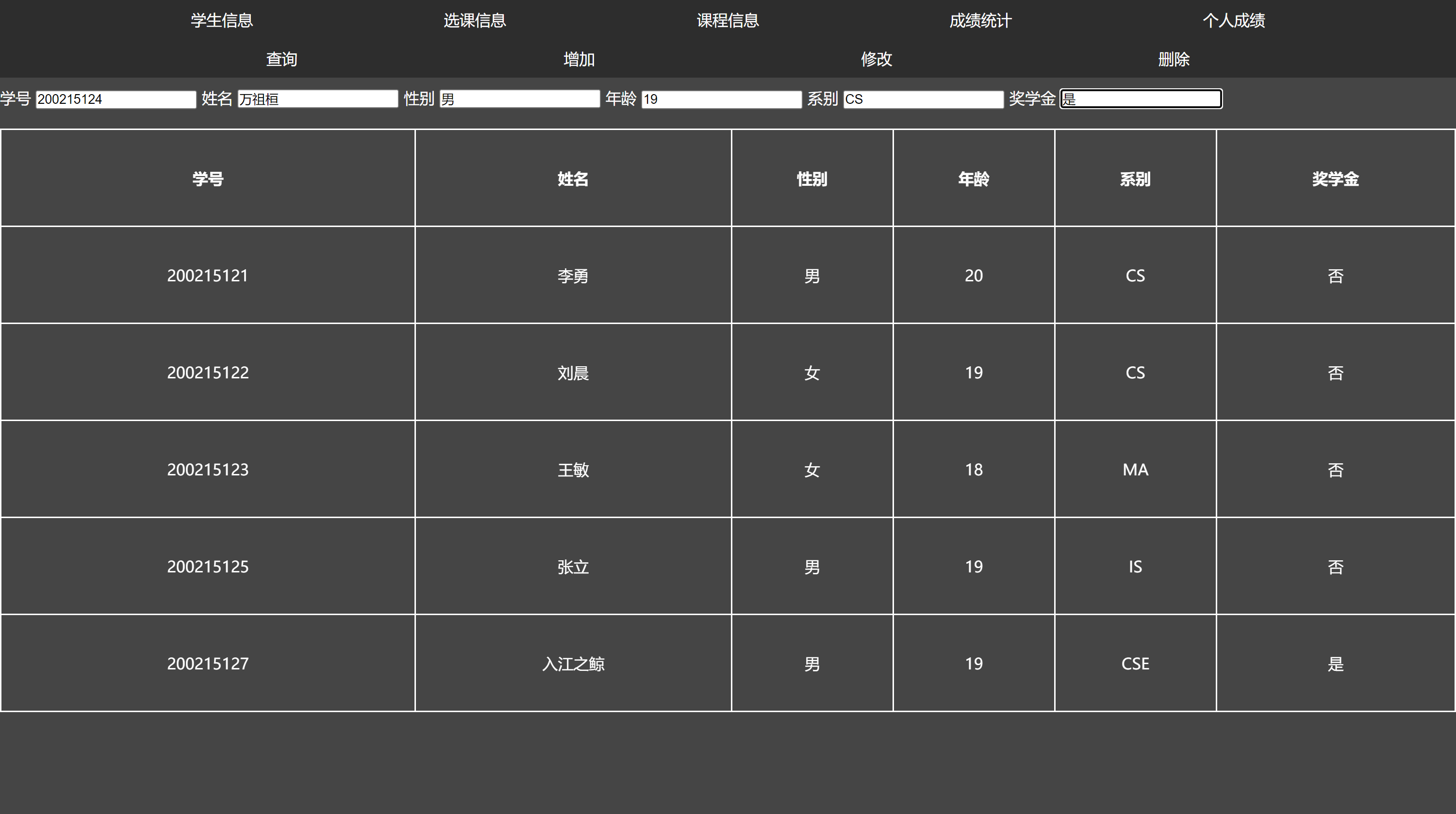
查询：点击查询按钮即可展示学生的信息：



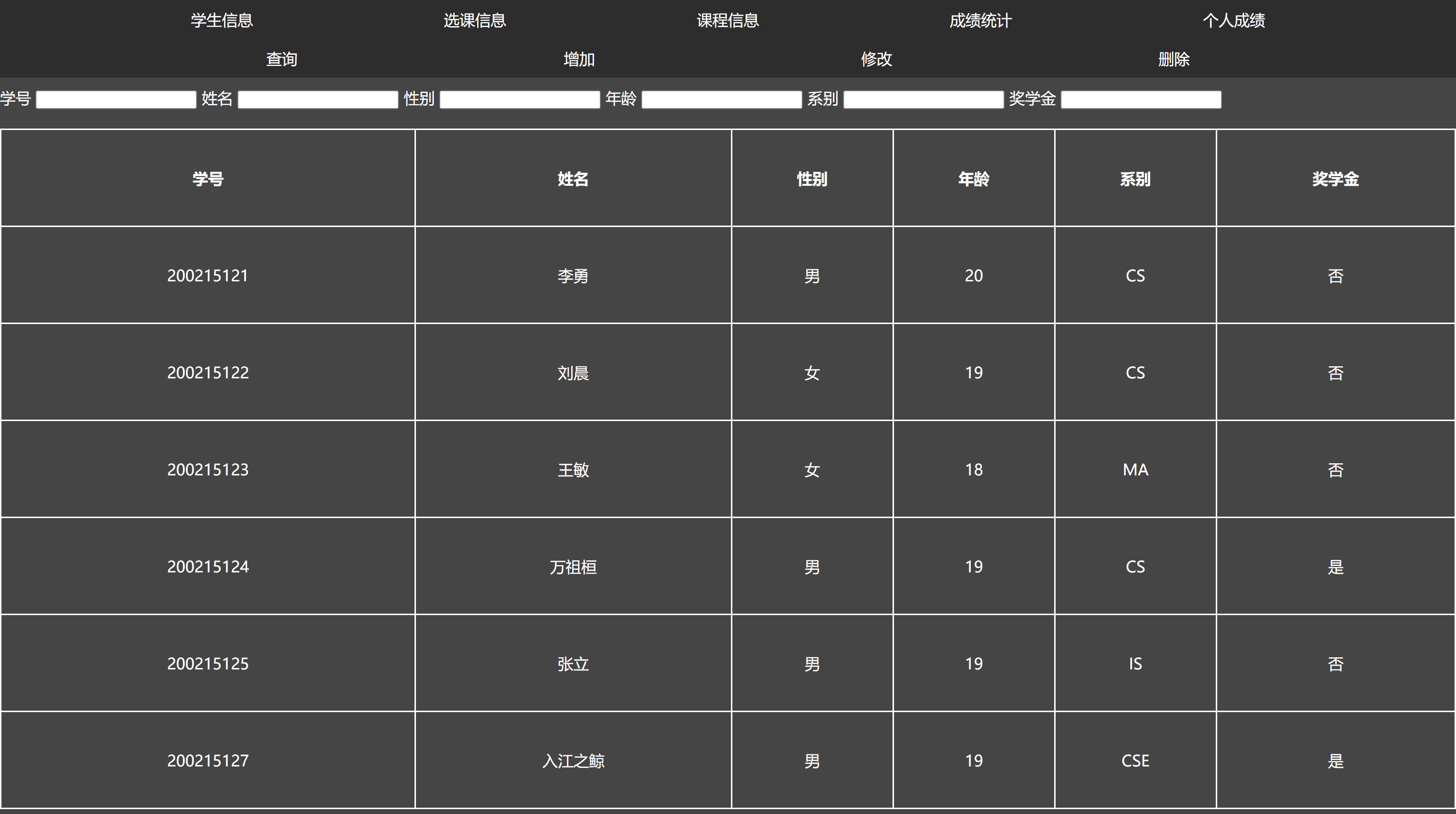
增加：

在输入框中输入学生的完整信息，点击增加按钮，即可向表中增加一名新生的信息。

增加前：



增加后：



修改：

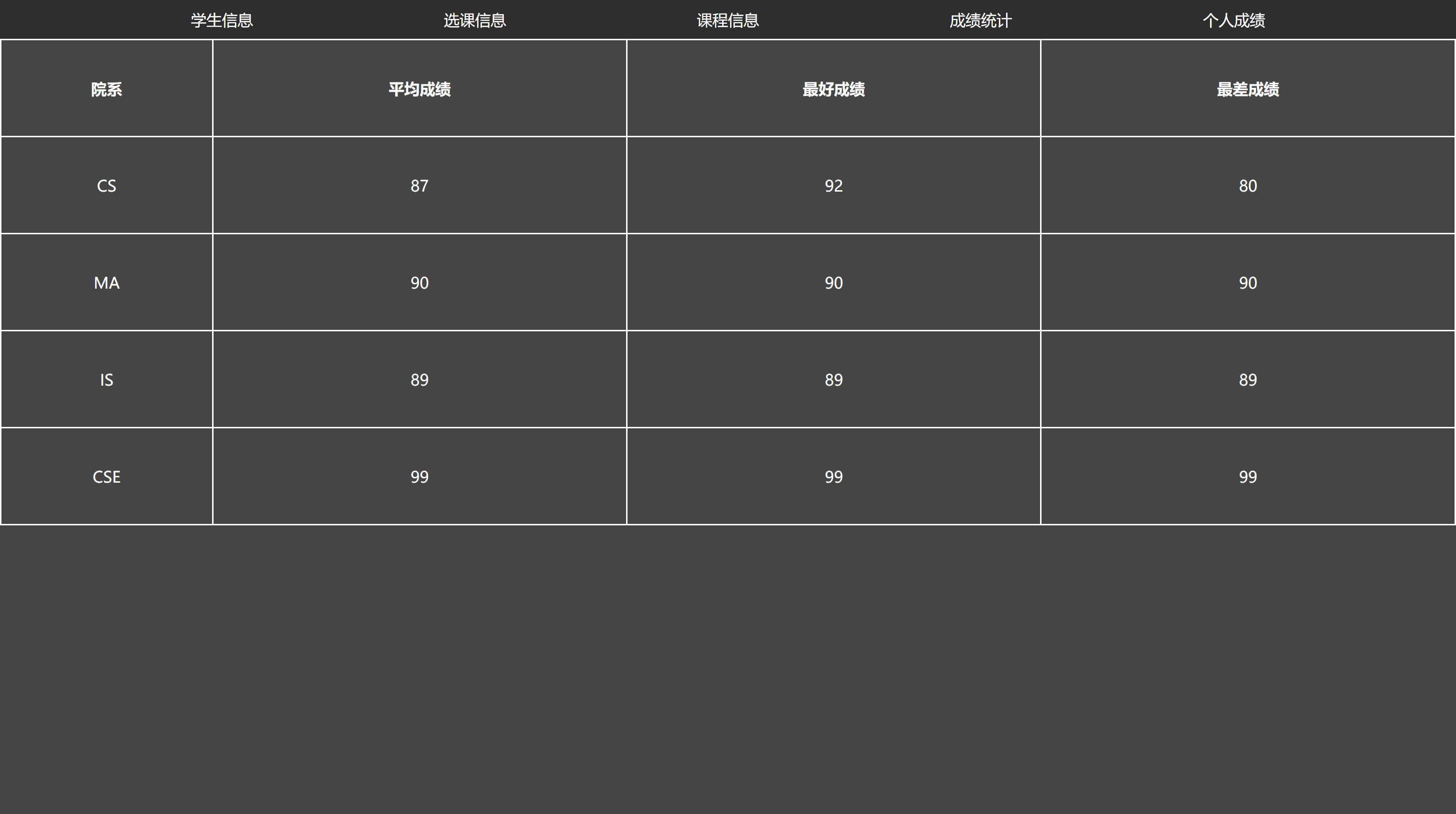
在输入框输入修改后的学生信息（修改的学生必须已在表中），点击修改即可

删除：

在输入框输入学生完整正确的学生信息方可删除改学生的信息。

成绩统计：

切换到成绩统计界面即可查看各个院系的总体成绩：



个人成绩查询：

输入学生学号即可查询改学生的各项信息





## 5.3任务总结

# 6 课程总结

前三个实验是主要是对SQL基本语法的学习和实践，通过看书和使用搜索引擎基本掌握了SQL语法，锻炼了自己解决问题的能力。在拓展练习的过程中中不断的将知识付诸与实践，在动手编程的过程中锻炼了自己的代码能力。

第四个实验是设计一个学生选课成绩管理系统，我运用自己所学的前端知识，搭建了一个前端页面来提升用户的体验，后端通过node.js中的express框架来实现，最终完成了一个可用的学生选课管理系统。

在这个全栈项目的实现过程中，我遇到了浏览器本身针对安全而产生的跨域问题，在反复查阅资料后，通过引入npm中的cors模块来解决，在此过程中及学习了使用JavaScript在node环境下如何操作mysql数据库，也了解了一些浏览器的安全问题，同时复习了这学期计算机网络的相关知识。

# 附录

项目的GitHub地址：<https://github.com/bowl23/HUST-database-experiment>，包含项目的源代码和后续react.js重构版本。